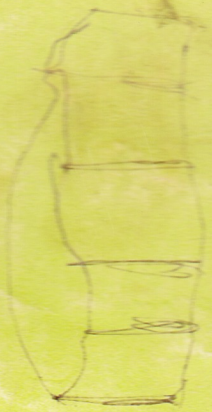


Manual de Basic Avanzado Para Computadores

versión atari basic

MANUAL DE BASIC AVANZADO PARA COMPUTADORES

VERSION ATARI BASIC

POR

COMPURAM

BASIC AVANZADO

Indice de Materias Pág.

- .- Nomenclatura. 1

1 .- Más sobre strings. 2

- Funciones ASC, CHR$, ESCAPE, Secuencias ESCAPE. 3-6
- Funciones LEN, STR$, VAL. 6-8
- Substrings. 9
- Manejo de substrings. 10
- Concatenamiento de strings. 10-11
- Strings numéricos. 11

2 .- Simulación de arreglos alfanuméricos. 12-13

3 .- Instrucciones:

POP, TRAP, LOCATE, CLR, COM. 14-17

4 .- Comandos de entrada/salida:

DOS, LIST, ENTER, SAVE, LOAD, LPRINT,
OPEN, CLOSE, INPUT#, PRINT#, PUT, GET,
NOTE, POINT, STATUS, XIO. 18-38

5 .- Funciones aritméticas, trigonométricas
y de uso especial:

ABS, CLOG, EXP, INT, LOG, RND, SGN, SQR,
ATN, COS, SIN, DEG, RAD, ADR, FRE, PEEK,
POKE, PADDLE, PTRIG, STICK, STRIG, USR. 39-52

6 .- La Unidad de Diskette 53

6.1 - Archivos en diskette. 54-55

6.2 - El Sistema Operativo de Disco
o D.O.S. (DOS). 56

6.3 - Partes del Dos. 57

6.4 - Transferencia de programas y datos. 58

6.5 - Uso de la Unidad de Diskette. 58-59

6.6 - El archivo MEM.SAV. 60

6.7 - Menú del DOS-OPCIONES. 60-73

6.8 - Programas en lenguaje Basic
en Diskette. 74

- Grabación de un programa. 74
- Carga de un programa. 75-76

NOMENCLATURA

vara : variable aritmética o numérica
vars : variable string
varm : variable matricial
var : variable. Puede ser cualquiera
opa : operador aritmético
opl : operador lógico
expa : expresión aritmética
expl : expresión lógica
exps : expresión string
exp : expresión Puede ser expa o exps
especarch : especificación de archivos
espec perif: especificación de periférico
dirmem : dirección de memoria
disp : dispositivo

Ejemplo:

Variable numérica: 1-2-3-4-415-21.428, etc.

Sus caracteres solamente pueden ser números. Se pueden hacer operaciones matemáticas con ellas.

Variable alfanumérica: A-B-11-OLA-HOLA-H-JAGUAR-YO-Z, etc.

Sus caracteres pueden ser números o letras. Cuando son números no tienen valor numérico o sea, no pueden sumar, multiplicar, etc.

Caracter: 1-A, 3;]. (* Z, etc.

Cada letra, número o signo es un caracter; incluso el espacio " " es un caracter.

BASIC AVANZADO

1.- MAS SOBRE STRINGS.-

En nuestro primer libro sobre BASIC, "Basic para Microcomputadores", definimos lo que es una variablle alfanumérica y su uso elemental en la elaboración de programas. Ahora veremos algunas técnicas de manejo de strings, que son de mucha utilidad.

El computador no almacena en memoria caracteres, sino únicamente números. Así todos los caracteres deben ser convertidos en números antes de ser guardados.

Los computadores usan un código, en el que a cada caracter se le asigna un número. La mayoría de los computadores usa el llamado código ASCII (American Standard Code for Information Interchange). ATARI usa un código levemente distinto, llamado ATASCII (ATARI ASCII). Así por ejemplo el código ATASCII para la letra A es 65, para la Z es 90, para el número 0 es 48, para el 9 es 57, etc.

FUNCION ASC.-

La función ASC convierte el primer caracter de una string al código ATASCII.

Escriba el siguiente programa en su pantalla:

```
10 DIM R$ (1)
20 ? "INGRESE UN CARACTER";: INPUT R$
30 ? "EL CODIGO ATASCII PARA   "; R$; "  ES  "; ASC(R$)
40 GOTO 20
```

Cada vez que escriba RUN y presione [Return] el programa pide un caracter (linea 20). Escriba una letra cualquiera y presione [RETURN].

```
10 PRINT CHR$(125):REM LIMPIA PANTALLA
20  DIM A$(7)
30 PRINT "INGRESE 7 CARACTERES";:INPUT A$
40 FOR I=1 TO 7
50 ?"EL CODIGO ATASCII PARA ";A$(I,I);" ES "
;ASC(A$(I,I))
60 NEXT I
```

6.9 - Archivos de Datos en Diskette. 76-78

6.10- Archivos secuenciales 78

6.10.1.- Apertura de archivo. 79

6.10.2.- Cierre de archivos de datos. 80

6.10.3.- Escribir en un archivo. 81-85

6.10.4.- Leer un archivo. 86-88

6.10.5.- Agregar información a archivos. 88-90

6.10.6.- Los números en los archivos. 91-93

6.11- Archivos de acceso directo (Random Access Files) 94-95

6.11.1.- Archivo indexado 95-96

6.11.2.- Ejemplo de archivo indexado 97-98

6.11.3.- Archivos indexados con DOS 3.0 98-99

FUNCION CHR$(N).-

Esta función devuelve el caracter correspondiente al número N del codigo ATASCII.

```
Escriba en la pantalla : PRINT CHR$ (65) Presione [RETURN]
                         A
Repita                 : PRINT CHR$ (48)
                         0
                         PRINT CHR$ (100)
                         d
```

```
10 PRINT CHR$(125)
20 FOR I=5 TO 15
30 A=INT(RND(0)*255)+1
40 POSITION 2,1:PRINT "PARA A= ";A
50 POSITION 15,I:PRINT "EL CARACTER ES:";CHR$(27);
CHR$(A)
60 NEXT I
```

ESCAPE (TECLA ESC.).-

Es un caracter (ASCII # 27) que se usa para generar un caracter distinto al que tiene asignado cada tecla, al presionarlo conjuntamente o previamente a la tecla o teclas que se va a digitar.

SECUENCIA EN QUE SE USA LA TECLA ESCAPE.-

Para poner caracteres especiales (por ejem. ↰, ↑, ↓, →, etc.) dentro de un string, existen las llamadas SECUENCIAS ESCAPE.

Si queremos asignar el caracter " ← " (movimiento del cursor) a una variable string, deberemos usar una secuencia ESCAPE. Proceda como sigue:

Presione la tecla [ESCAPE] y luego la combinación que genera el caracter deseado. En el caso que nos ocupa deberíamos digitar [ESC], soltarla, luego presionar [CTRL] y sin soltarla presionar [+]. El cursor no se moverá a la izquierda como habría sido lo normal si no hubieramos presionado [ESC] primero.

Lo que se ve al digitar una secuencia ESCAPE no es exactamente lo que se obtiene como string.

Veamos el caso siguiente:

(digitar [ESC] - [CTRL] [+] para ← y [ESC] - [CTRL] [=] para ↓)

a) PRINT "HOLA ATA←←↓ RI" (No olvide el espacio)
Se obtendrá:

HOLA ATA
RI

Otro ejemplo:

b) PRINT "→ →"; "HOLA"
Se obtendrá HOLA, corrido dos espacios hacia la derecha.

Las secuencia ESCAPE son:

COMBINACION DE TECLAS	CARACTER MOSTRADO	CODIGO ATASCII	SIGNIFICADO DE LA STRING
[ESC]-[ESC]		27	Código escape
[ESC]-[BACKS]		126	Cursor 1 lugar a la izquierda y reemplaza por espacio en blanco.
[ESC]-[TAB]		127	Cursor a la derecha hasta el TAB stop siguiente.
[ESC]-[CTRL][-]		28	Cursor 1 lugar hacia arriba.
[ESC]-[CTRL][=]		29	Cursor 1 lugar hacia abajo.
[ESC]-[CTRL][*]		30	Cursor 1 lugar hacia la derecha.
[ESC]-[CTRL][+]		31	Cursor 1 lugar hacia la izquierda.
[ESC]-[CTRL][BACKS]		254	Borra caracter
[ESC]-[CTRL][>]		255	Inserta caracter
[ESC]-[CTRL][<]		125	Limpia pantalla
[ESC]-[CTRL][TAB]		158	Anula los "stop" del TAB
[ESC]-[CTRL][2]		253	Sonido por el parlante
[ESC]-[SHIFT][BACKS]		156	Borra línea
[ESC]-[SHIFT][>]		157	Inserta línea
[ESC]-[SHIFT][<]		125	Limpia pantalla
[ESC]-[SHIFT][TAB]		159	Coloca los "stop" del TAB.

TABLA 1-1

Cualquiera de estos caracteres que aparecen en la columna "caracter mostrado" es posible presentarlos en pantalla. Para ello digite [ESC]-[ESC] antes de la secuencia ESCAPE deseada.

Por ejemplo: si desea obtener en pantalla , debe digitar:

```
PRINT "[ESC]-[ESC]-[ESC]-[TAB]"
```

Obtendrá ▶

```
10 PRINT CHR$(125):POSITION 2,11
20 PRINT "(6 VECES [ESC]-[CTRL] [INSERT]) ATARI"";
30 PRINT "[ESC]-[CTRL] [-]"
40 FOR I=1 TO 40:NEXT I
50 GOTO 20
```

Otras funciones que se refieren a strings son:

a) LEN: entrega un número decimal que representa el largo de una string.
Escriba en la pantalla:

```
5 DIM A$(10)                    [RETURN]
10 A$="ROMA"                       "
20 A=LEN (A$)                      "
30 PRINT "A="; A                   "
RUN                                "
```

Resulta A=4, que es el largo de la string ROMA.

Cambie en la línea 10 la palabra "ROMA" por otras a su elección y observe los resultados.

b) STR$: convierte un valor numérico en un string.
Ejemplo:

```
10 DIM A$(2): A=25
20 A$=STR$ (A)
30 PRINT "A$="; A$
```

Resulta A$=25. La expresión .25 ya no es un valor numérico. Es un string numérico. (No se pueden hacer operaciones matemáticas con él).

c) VAL: Convierte un string numérico en un valor numérico.

Ejemplo: al programa anterior agregue ahora:

```
40 B=VAL (A$)
50 PRINT "B="; B
```

Resultará B=25. Ahora la expresión 25 ha vuelto a ser un valor numérico y como tal puede operarse matemáticamente con el.

```
10 REM NUMEROS PALINDROMICOS
20 REM PALINDROMICOS SON LOS
30 REM NUMEROS QUE LEIDOS HACIA
40 REM LA DERECHA O IZQUIERDA
50 REM SON IGUALES EJ: 1221
60 DIM A$(4),B$(4)
65 PRINT CHR$(125):REM LIMPIA PANTALLA
70 FOR I=1 TO 9999
80 A$=STR$(I)
90 IF LEN(A$)=4 THEN GOTO 110
100 A$=STR$(I):A$(LEN(A$)+1,4="
110 B$(1,1)=A$(4,4)
120 B$(2,2)=A$(3,3)
130 B$(3,3)=A$(2,2)
140 B$(4,4)=A$(1,1)
150 IF VAL(A$)<>VAL(B$) THEN 170
160 ?"EL ";I;"ES PALINDROMICO"
170 NEXT I
```

```
10 REM VALIDACION FECHA
20 PRINT CHR$(125)
30 DIM A$(8)
40 A1=1900:A2=2000:REM FECHAS TOPES
50 PRINT "INGRESE LA FECHA DD MM AAAA ";:INPUT A$
60 D=VAL(A$(1,2)):M=VAL(A$(3,4')):A=VAL(A$(5,8))
65 IF A <A1 OR A>A2 THEN GOSUB 9000:GOTO 50
70 IF M=2 THEN 1000
80 IF M=1 OR M=3 OR M=5 OR M=7 OR M=8 OR  M=10  OR  M=12
   THEN 2000
90 IF M=4 OR M=6 OR M=9 OR M=11 THEN 3000
100 GOSUB 5000:GOTO 50
1000 IF D>28 THEN GOSUB 6000:GOTO 50
1010 GOTO 12000
2000 IF D>31 THEN GOSUB 7000:GOTO 50
2010 GOTO 12000
3000 IF D>30 THEN GOSUB 8000:GOTO 50
3010 GOTO 12000
5000 ? "MES INVALIDO > 12 ":RETURN
6000 ? " DIA INVALIDO > 28 ":RETURN
7000 ? " DIA INVALIDO > 31 ":RETURN
8000 ? " DIA INVALIDO > 30 ":RETURN
9000 ? " A#O INVALIDO ":RETURN
12000 ? " CORRECTO "
```

SUBSTRINGS.-

Los SUBSTRINGS son porciones de una variable alfanumérica. Para designar un SUBSTRING debe indicarse el nombre de la variable alfanumérica y seguidamente, entre paréntesis, los números correspondientes a la posición del 1er. y del último caracter del substring que se desea separar.

Por ejemplo:

Si A$="PARALELEPIPEDO", A$(5,9) designará el substring LELEP: desde el 5to. caracter hasta el 9o.

Los SUBSTRINGS pudieran parecer como elementos de un arreglo, pero no lo son. ATARI BASIC no acepta arreglos de variables alfanuméricas. Es innecesario especificar la posición del último caracter si se desea especificar toda la porción derecha del string. Por ejemplo: si se desea especificar como SUBSTRING la expresión PIPEDO, bastará con indicarla como A$(9).

El primero y el último caracter del substring pueden ser especificados mediante constantes, variables o expresiones numéricas.

Cualquier problema que se presente con respecto a la especificación del substring dará como resultado ERROR 5. El último caracter del substring no puede tener valor inferior al del primero. Por ejemplo: si se especifica: A$ (6,2) o A$ (9,1), el computador indicará ERROR 5.

Tampoco está permitido especificar el número 0 (cero) como primer o último caracter.

MANEJO DE SUBSTRINGS

```
10 PRINT CHR$(125)
20 DIM NOMBRE$(250),NO$(25),R$(1)
30 REM LIMPIA LOS STRINGS
40 NOMBRE$="   ":NOMBRE$(250=NOMBRE$:NOMBRE$(2)=NOMBRE$
50 REM INGRESO DE DATOS
60 FOR K=1 TO 10
70 PRINT "INGRESE NOMBRE NRO. ";K;"   ";:INPUT NO$
80 NOMBRE$((K*25)-24,K*25)=NO$
90 NEXT K
100 REM LECTURA DE LOS DATOS
110 PRINT CHR$(125)
120 PRINT "INGRESE UN NUMERO ENTRE 1 Y 10";:INPUT N
130 PRINT "EL ELEMENTO NRO.";N;" DE LA LISTA ES ";
    NOMBRE$((N*25)-24,N*25)
140 PRINT "DESEA SEGUIR (S/N)";:INPUT R$
150 IF R$="N" THEN END
160 GOTO 110
```

CONCATENAMIENTO DE STRINGS.-

Es posible unir strings para formar un string más largo. Esto se llama CONCATENAMIENTO DE STRINGS. Con el CONCATENAMIENTO es posible desarrollar strings de cualquier largo. Sólo se está limitado por la cantidad de memoria RAM, disponible.

La función LEN permite concatenar strings en ATARI BASIC. LEN es una función que entrega el número de bytes que ocupa la variable alfanumérica sobre la que es aplicada.

Ejemplo:

```
10 DIM J$ (8)
20 J$ = "COMPURAM"
30 ? LEN (J$)
```

Si corre el programa obtendrá como resultado 8.

Veamos ahora un resultado de concatenación:

```
10 DIM A$(8), B$(9), C$(5)
20 A$= "CASA": B$= "MATA": C$= "DORES"
30 A$ (LEN(A$) + 1)= B$
40 B$ (LEN (B$) + 1)= C$
50 PRINT A$, B$
```

```
10 PRINT CHR$(125)
20 DIM A$(20)
30 PRINT "INGRESE ALGUNOS CARACTERES (MAX 20)"
40 INPUT A$
50 PRINT "UD. INGRESO ";LEN(A$);" CARACTERES"
```

STRINGS NUMERICOS.-

Un string numérico es aquel cuyo contenido puede ser evaluado como un número. Los valores numéricos pueden ser convertidos en valores alfanuméricos usando la función STR$.

La regla general es que el valor numérico es convertido a valor alfanumérico en la misma forma en que aparecería si fuera tratado por una instrucción PRINT. Por ejemplo: si el valor numérico es un número negativo, entonces el primer caracter del valor alfanumérico será el signo menos. O bien si el valor numérico es un valor muy grande o muy chico, será expresado en notación científica al ser convertido en un valor alfanumérico.

Ilustremos esta situación con un ejemplo:

```
10 DIM R$ (10)
20 ? "INGRESE UN NUMERO",: INPUT R
30 R$ = STR$ (R)
40 ? "VALOR NUMERICO:"; R
50 ? "VALOR ALFANUMERICO:"; R$
60 GOTO 20
```

Corra el programa e ingrese algunos números positivos y algunos negativos, algunos números grandes (atención al DIM!) y algunos pequeños.

Para concatenar strings numéricos puede usarse la función LEN. Digite el siguiente programa:

```
10 DIM K$ (50)
20 ? "INGRESE UNOS POCOS DIGITOS"
30 INPUT N
40 K$ (LEN (K$) + 1) = STR$ (N)
50 ? "EL NUEVO NUMERO ES :"; K$: ?
60 GOTO 20
```

2.- SIMULACION DE ARREGLOS ALFANUMERICOS.-

ATARI BASIC no tiene la facultad de trabajar con arreglos alfanuméricos.

Puede simularse un arreglo alfanumérico usando una variable alfanumérica. Esta situación dá origen a un PSEUDO-ARREGLO.

La simulación consiste en dividir el string en sub trings de igual largo y tratar cada substring como un elemento del pseudo-arreglo.

Para determinar la posición de un elemento del pseudo-arreglo es preciso conocer el número de orden del elemento y su longitud.

Hay dos limitaciones concernientes a los pseudo-arreglos alfanuméricos:

1.- Todos los substrings deben tener el mismo largo. El largo debe ser el máximo. Si hay elementos en que el texto es más corto, los espacios sobrantes deben llenarse

2.- El proceso de cálculo de la ubicación del comienzo de cada elemento del pseudo-arreglo es lento.

Consideremos un ejemplo en que se use un pseudo-arreglo alfanumérico.

```
10 DIM A$(100), R$(10), BL$(10)
20 DATA PEDRO, ISABEL, RICARDO, MONICA, CAROLINA, ****,
   123456789012, , PARALELEPIPEDO, A*B/C:REM 10 VALORES
   ALFANUMERICO
30 BL$="    ": BL$(10)=BL$:BL$(2)= BL$: REM INICIALIZA LOS
   ELEMENTOS CON ESPACIOS EN BLANCO
40 FOR K=1 TO 10
50 READ R$
60 LARGO = LEN (R$)
70 IF LARGO < 10 THEN R$ (LARGO + 1)= BL$: REM PONE ESPACIOS
   EN BLANCO EN ESPACIOS SOBRANTES DE CADA SUBSTRING
80 PRINCIPIO = (K-1) * 10 + 1
90 A$ (PRINCIPIO) = R$
100 NEXT K
110 FOR K = 1TO10:REM MUESTRA VALORES ASIGNADOS
115 PRINCIPIO = (K - 1) * 10 + 1
120 ? "ELEMENTO"; K;" ES: "; A$ (PRINCIPIO, PRINCIPIO + 9)
130 NEXT K
140 END
```

En este programa la variable alfanumérica A$ hace de anfitrión del pseudo-arreglo. Este tiene 10 elementos de 10 caracteres cada uno, para un total de 100 caractere (línea 10).

A cada elemento corresponde uno de los valores alfanuméricos de la DATA (línea 20). Entre éstos hay un valor espacio vacío y 2 valores que tienen más de 10 caracteres.

El programa asigna a la variable BL$ un valor espacio vacío (línea 30), usa una variable auxiliar, R$ (línea 50) para leer los DATA. Esto es necesario pués ATARI BASIC no permite variables indexadas en la instrucción READ.

En línea 60 y 70 el programa concatena espacios blancos con los caracteres correspondientes de R$

Finalmente, hace un listado de los elementos del pseudo-arreglo y los correspondientes valores provenientes de la DATA.

3.- INSTRUCCIONES POP, TRAP, LOCATE, CLR, COM.-

POP

Formato: POP

Es una instrucción ligada en cierta forma a GOSUB. Si se desea que una subrutina NO vuelva a la instrucción siguiente a GOSUB, podría usarse una instrucción GOTO para volver al lugar deseado del programa principal; sin embargo, el BASIC todavía "recuerda" donde debería haber vuelto. En tales casos debe usarse la instrucción POP. Todo lo que POP hace es que el BASIC olvide el lugar donde debería volver después de ejecutar una subrutina.

La instrucción POP debe usarse de acuerdo a las siguientes reglas:

1.- Debe estar en la trayectoria de ejecución del programa principal.

2.- Debe seguir a la ejecución de cualquiera instrucción GOSUB que no vuelve al programa principal mediante una instrucción RETURN.

El ejemplo a continuación ilustra el uso de POP.

```
10 REM USO DE LA INSTRUCCION POP
20 REM EJECUTE Y LUEGO AGREGUE LA LINEA "5035 POP"
30 PRINT CHR$(125)
40 PRINT "LLAMADA DE LA PRIMERA SUBRUTINA"
50 GOSUB 5000
60 PRINT "VUELTA A PROGRAMA PRINCIPAL"
70 PRINT "FIN"
80 END
5000 PRINT "PRIMERA SUBRUTINA EN EJECUCION"
5010 PRINT "LLAMADA A LA SEGUNDA SUBRUTINA"
5020 GOSUB 6000
5030 PRINT "REGRESO A LA PRIMERA SUBRUTINA"
5040 RETURN
6000 PRINT "SEGUNDA SUBRUTINA EN EJECUCION"
6010 PRINT "LA RAIZ CUADRADA DE 2 ES";SQR(2)
6020 GOTO 5030
```

Corra el programa.- Puede verse que la frase REGRESO A LA PRIMERA SUBRUTINA se repite 2 veces antes de volver al programa principal. Agregue la línea 5035 POP y corra el programa nuevamente. La instrucción POP hace olvidar la instrucción de vuelta para la 2da. subrutina, de modo que al encontrar el RETURN (línea 5040), vuelve a la línea 60.

TRAP (T.)

Formato: TRAP expa

Esta es una instrucción que se usa para desviar la ejecución de un programa a una línea especificada si se detecta un error.

La instrucción TRAP impide que se detenga la ejecución del programa y se muestre en pantalla un mensaje de error.

TRAP funciona con cualquier error que se produzca una vez que ha sido ejecutada la línea del programa, pero sirve una sola vez y debe reactivarse. En consecuencia es aconsejable colocar el TRAP al comienzo del sector de programa en que se prevée la posible ocurrencia de un error. De esta manera al producirse el error volvera al comienzo del sector mencionado y la instrucción (TRAP) se reactivará sola.

```
10 REM DEMOSTRATIVO DE TRAP
20 TRAP 20
30 PRINT "PARA CONTINUAR PRESIONE [RETURN]"
40 INPUT A
50 END
```

En este caso el error se produce porque al presionar RETURN se ingresa un caracter alfanumérico y la variable A es numérica solamente.

El uso de TRAP puede dificultar la revisión de programas con errores, ya que el programa se desviará automáticamente hasta la línea de programa del último TRAP ejecutado.

```
10 REM USO DE LA INSTRUCCION TRAP
20 REM INGRESE VALORES <=0
30 PRINT CHR$(125)
40 TRAP 2000
50 PRINT "INGRESE UN NUMERO";:INPUT A
60 RES=LOG(A)
70 PRINT "EL LOGARITMO NATURAL DE";A:PRINT"ES";LOG(A)
80 END
2000 PRINT :PRINT "HAY UN ERROR CODIGO";PEEK(195)
2010 LIN=PEEK(186)+PEEK(187)*256
2020 PRINT "EN LA LINEA NRO.";LIN
2030 LIST LIN
2040 FOR R=1 TO 2000:NEXT R
2050 GOTO 30
```

LOCATE (LOC.)

Formato: LOCATE col, fila, vara:

Esta instrucción entrega el código del caracter ubicado en las coordenadas especificadas. En el caso de modos gráficos puros, entregará el valor correspondiente a la función COLOR usada.

LOCATE dará:

a) números entre 0 y 255 para los modos de texto
b) 0 o 1 para los modos gráficos de 2 colores
c) 0, 1, 2 o 3 para los modos gráficos de 4 colores.

Para que la instrucción LOCATE trabaje, el canal #6 debe estar abierto para poder presentar información en pantalla. La instrucción GRAPHICS, abre dicho canal.

La instrucción LOCATE es equivalente a:

POSITION col, fila: GET #6, vara.

La ejecución de una instrucción PRINT después de LOCATE puede cambiar el código del caracter o punto gráfico recién examinado, con lo que se echa a perder lo mostrado en pantalla. En tal caso, para solucionar el problema reubique el cursor (con POSITION o PLOT) y coloque de nuevo el dato (con PUT #6, vara) antes de efectuar el PRINT.

```
10 REM USO DE LOCATE
20 GR. 3 + 6
30 SE. 2,11,10:COLOR 3
40 PL.5,9:DR.35,9
45 FORD=1 TO 500: NEXT D
50 LOCATE 20,9, X
60 PRINT "VALOR DE LA
   FUNCION COLOR ASOCIADA
   AL PIXEL UBICADO EN
   COL 20, LINEA 9"; X
```

```
10 REM USO DE LOCATE
20 GR. 0
30 POS.15,10:PRINT "ATARI"
35 FORD=1 TO 500:NEXT D
40 LOCATE 17,10, X
50 POS. 17,12

60 PRINT "CARACTER UBICADO EN
   COL 17, LINEA 10 ";X
```

CLR

Formato: CLR

Este comando permite:

- poner todas las variables numéricas a 0
- desdimensionar las variables alfanuméricas y los arreglos numéricos
- coloca el puntero al comienzo de la lista de valores de una instrucción DATA

Después de ejecutar una instrucción CLR, es posible redimensionar arreglos y variables alfanuméricas. Sin embargo CLR no borra nombres de variables de la tabla de nombres de variables, de modo que no es posible utilizarlo para hacer espacio para nuevos nombres de variables en caso de que en un programa se haya rebasado el límite de 128 nombres.

El único comando que permite borrar nombres de la tabla de nombres de variables es NEW. (Que borra además el programa).

COM

Formato: COM vars (expa)
COM vara (expa1, expa2)

Este comando reserva espacio en memoria para el manejo de arreglos numéricos y variables alfanuméricas.

Es exactamente igual al comando DIM.

4.- COMANDOS DE ENTRADA/SALIDA. (INPUT/OUTPUT)

DOS (DO.)

Formato: DOS

Este comando activa el Menú de Utilitarios del sistema operativo de discos. Si el sistema operativo de disco (DOS) no ha sido cargado en memoria el computador pasará al modo MEMO PAD (línea 400/800) o al modo SELF/TEST (línea XL). En tal caso presionando [SYSTEM RESET] se volverá al modo DIRECTO (el control del computador lo tiene el lenguaje BASIC). Si el DOS ha sido cargado previamente, el hecho de escribir DOS en pantalla bará aparecer el Menú de Utilitarios del DOS.

Para limpiar la pantalla del Menú del DOS puede presionarse [SYSTEM RESET]; o bien, seleccionar la opción RUN CARTRIDGE (DOS 2.0S) o TO CARTRIDGE (DOS 3.0) del Menú. Cualquiera de estas acciones transfiere el control al lenguaje BASIC.

El comando DOS se debe usar en modo directo.

Cuando se dá una instrucción DOS el computador:

- limpia la pantalla
- repone en los registros de color los colores normales o default
- corta todos los registros de sonido
- cierra todos los canales de entrada/salida, excepto el canal 0.
- muestra en pantalla el Menú del DOS correspondiente

LIST (L.)

Formato: LIST especarch, # línea 1, # línea 2

Transfiere un programa BASIC almacenado en memoria a un dispositivo de grabación: cassette, diskette o, impresora.

Para grabar en cassette debe usarse:

LIST "C:", 200, 1000 [RETURN]

Para grabar en diskette debe usarse:

LIST "D:" NOMBARCH. BAS, 200, 1000 [RETURN]

De esta manera el programa será grabado en cassette o diskette, desde la línea 200 hasta la 1.000. En el caso de diskette, debe darse un nombre al archivo (máx. 8 caracteres). Si no se especifican líneas el programa será grabado integramente.

LIST "D:NOMBARCH" [RETURN].

LIST envía el texto BASIC en código ATASCII, sea cual sea el equipo periférico destinatario que se use.

Los comandos CSAVE y SAVE no pueden leer las grabaciones realizadas usando LIST.

El comando ENTER permite leer los programas grabados usando LIST.

Al grabar un programa BASIC, el comando LIST no graba la tabla de nombres de variables.

El comando LIST "P:", escribirá en la impresora todo el listado de programa que se encuentre en la memoria del computador, a través del canal #7.

ENTER (E.)

Formato: ENTER especarch, # línea 1, # línea 2

El comando ENTER permite transferir o cargar en la memoria un programa BASIC, que haya sido grabado previamente en cassette o diskette usando el comando LIST.

Para cargar desde cassette debe usarse:
ENTER "C:", 20, 1000

Para cargar desde diskette debe usarse:

ENTER "D: NOMBARCH.BAS"

Si se especifican líneas se cargará en memoria la parte de programa especificada por las líneas.

El comando ENTER, es parecido a los comandos CLOAD y LOAD, pero existen diferencias importantes.

ENTER no borra las líneas de programa existentes en memoria antes de transferir nuevas líneas, sino que las agrega a continuación de las existentes. Si la numeración de las líneas es igual, entonces las nuevas líneas reemplazarán a las antiguas.

ENTER sólo puede transferir texto en BASIC que haya sido grabado en código ATASCII, de modo que trabaja sólo con programas que hayan sido grabados usando el comando LIST. No trabaja con programas grabados mediante el uso de CSAVE o SAVE.

El comando ENTER, al encontrar un nombre de variable nuevo en el programa que se está ingresando a memoria lo agrega a la lista de nombres de variables. ENTER no quita ningún nombre de la lista de nombres de variables.

SAVE (S.)

Formato: SAVE especarch

El comando SAVE permite transferir un programa en BASIC desde la memoria del computador a un dispositivo de grabación: cassette o diskette.

Para transferir a cassette debe usarse:

SAVE "C:"

Para transferir a diskette debe usarse:

SAVE "D: NOMBARCH.BAS"

SAVE graba en cassette o diskette, la versión en código simbólico (tokenized) del programa en BASIC y sólo los comandos LOAD Y RUN pueden ser empleados para realizar la operación inversa, es decir, cargar el programa desde cassette o diskette a memoria del computador.

Los comandos CLOAD y ENTER no pueden cargar un programa que haya sido grabado con SAVE.

Durante el proceso de grabación el comando SAVE graba la tabla de nombres de variables junto con el programa.

LOAD (LO.)

Formato: LOAD especarch

LOAD permite cargar en memoria un programa en BASIC, que haya sido previamente grabado mediante el uso del comando SAVE.

Para cargar desde cassette:

LOAD "C:"

Para cargar desde diskette:

LOAD "D: NOMBARCH.BAS"

El uso del comando LOAD, automáticamente involucra el comando NEW. Limpia la memoria del computador de todo posible programa que hubiere estado allí.

El comando LOAD sólo puede cargar programas que hayan sido grabados con SAVE.

LPRINT (LP.)

Formato: LPRINT exp (;,) exp

Ejemplo: LPRINT A; R$,"VELOCIDAD="; VELO

Este comando es semejante al comando PRINT, excepto que la información pasa a la impresora en vez de ir a la pantalla.

LPRINT usa el canal 7 para enviar información a la impresora. Si el canal 7 está abierto hacia otro periférico, se producirá un error que cerrará el canal.

OPEN (O.)

Formato: OPEN # canal, expa 1, expa 2, espec perif.

Ejemplos:

```
100 OPEN #1, 4, 0, "C:"
215 OPEN #3, 8, 0, "P:"
520 OPEN #5, 12, 0, "D:PROGRAM2.BAS"
```

El comando o instrucción OPEN permite asignar un canal de entrada/salida de información a un equipo periférico, incluyendo en esta denominación un archivo en disco.

Se debe abrir un canal antes de que el BASIC pueda tener acceso a un periférico para traer o enviar información (input/output). Si el canal ya está abierto hacia otro dispositivo que el deseado, se producirá un error.

El primer parámetro del comando OPEN, # canal, especifica el canal por el cual se producirá el flujo de información.

El computador ATARI se comunica con todos los equipos periféricos de entrada/salida mediante los canales de entrada/salida. Sinónimo del término canal es IOCB (input/output control blocks).

Hay 8 canales en total, numerados del 0 al 7. BASIC reserva los canales 0, 6 y 7 para actividades específicas.

El canal 0 está permanentemente reservado para el editor (dispositivo E:). Las instrucciones simples INPUT y PRINT usan el canal 0.

El canal 6 es usado por las instrucciones del ATARI BASIC relacionadas con gráficos presentados en pantalla.

El canal 7 se usa para algunas operaciones de la impresora y para la grabación y carga de programas.

Los canales 1 al 5 están completamente disponibles para ser usados con programas en BASIC. Los canales 6 y 7 pueden ser usados con limitaciones. Si el programa no usa instrucciones relativas a gráficos puede usarse el canal 6. Si no hay instrucciones relativas a grabar (SAVE) o cargar (LOAD) ni se usa la instrucción LPRINT, puede usarse el canal 7.

El segundo parámetro de la instrucción OPEN especifica el tipo de acción que se desarrollará por el canal. Esta acción puede ser entrada de datos, salida de datos o ambas.

La tabla 4.1 a continuación especifica y ayuda a comprender el uso del segundo parámetro

Periférico	2o. Param. (expa1)	Descripción de la acción
Cassetera (C:)	4	Leer
	8	Escribir
U. Diskette (D[n]:NOMBARCH.EXT)	4	Leer un archivo existente
	6	Leer directorio de diskette
	8	Escribir - archivo nuevo
	9	Escribir - agregar registros al final del archivo existente
	12	Leer y escribir - actualizar archivo existente
	13	Leer, escribir o agregar registros a archivo existente
Editor de pantalla (E:)	8	Salida a pantalla
	12	Entrada por teclado y salida a pantalla
	13	Entrada y salida de pantalla
Teclado (K:)	4	Leer
Impresora (P:)	8	Escribir
RS-232 puerta serie (R[n]:)	5	Lectura concurrente
	8	Escritura en bloque
	9	Escritura concurrente
	13	Lectura y escritura concurrente

		Limpia Pantalla	Ventana Texto (2)	Leer	Escrit
Pantalla (S:)	8	si	no	no	si
	12	si	no	si	si
	24	si	si	no	si
	28	si	si	si	si
	40	no (1)	no	no	si
	44	no (1)	no	si	si
	56	no (1)	si	no	si
	60	no (1)	si	si	si

(1) En modo gráfico 0 la pantalla es siempre limpiada
(2) En modo gráfico 0 no hay ventana de texto separada

TABLA 4-1
2do. Parámetro en OPEN: expa1

El 3er. parámetro del comando OPEN es usado en raras ocasiones. La mayoría de las veces toma el valor 0.

La tabla 4-2, muestra los valores y su uso.

Periférico	3er.Param.	Descripción de la acción
Cassettera (C:)	0	Espaciamiento normal de grabación
	128	Espaciamiento corto de grabación
U. Diskette (D[n]:NOMBARCH.EXT)	0	Ignorada
Editor de pantalla (E:)	0	Ignorada
Teclado (K:)	0	Ignorada
Impresora (P:)	0	Caracteres normales
RS-232 puerta serie (R[n]:)	0	Ignorada
Pantalla (S:)	0	Modo gráfico 0. BASIC
	1	" " 1. BASIC
	2	" " 2. BASIC
	3	" " 3. BASIC
	4	" " 4. BASIC
	5	" " 5. BASIC
	6	" " 6. BASIC
	7	" " 7. BASIC
	8	" " 8. BASIC

TABLA 4-2

3er. Parámetro en OPEN: expa2

El 4to. parámetro de la instrucción OPEN, espec perif, señala el equipo periférico al cual el canal estará asociado. La tabla 4-3, a continuación, indica los nombres standarizados para los periféricos.

Periférico	Nombre
Cassettera	C:
U. Diskette	D[n]:NOMBARCH.EXT
Editor de pantalla	E:
Teclado	K:
Impresora	P:
RS-232 puerta serie	R[n]:
Pantalla	S:

TABLA 4-3

Nombres de periféricos en OPEN

CLOSE (CL.)

Formato: CLOSE #1
CL. # CARTA

El comando CLOSE cierra un canal de entrada/salida de información a un equipo periférico.

Un canal que esté abierto (con OPEN) para entrada, salida o ambas, debe ser previamente cerrado antes de que pueda ser reasignado a otro periférico diferente. Cerrar un canal que no ha sido abierto no tiene ningún efecto. No se produce ningún error.

El comando END cierra todos los canales abiertos excepto el canal 0, que BASIC usa para la comunicación normal o habitual (standard) con el teclado y la pantalla.

INPUT

Formato: INPUT #2, J, K, L, M
INPUT #3, AUTO$, RAMO$

La instrucción INPUT saca una línea de datos desde un periférico de entrada. La línea de datos consiste de 0 o más caracteres ATASCII seguidos por un caracter ATASCII correspondiente a "TERMINO DE LINEA" (EOL).

En el teclado, la tecla [RETURN] produce un caracter EOL al final de la línea ingresada. Cualquiera sea el periférico de entrada se requiere el caracter EOL para dar por ingresada una línea de datos.

El computador interpreta la línea de datos como un valor alfanumérico, uno o más valores numéricos o alguna combinación de éstos.

La manera en que la interpreta depende del tipo y número de variables de que se trate (numéricas y/o alfanuméricas), pero es totalmente independiente del tipo de periférico de entrada.

Si la opción "# canal" no aparece en la instrucción, el INPUT proviene del teclado vía el editor (dispositivo E:). Si el "# canal" aparece, éste debe ser abierto para una entrada previamente mediante la instrucción OPEN.

```
10 REM USO DE LA INSTRUCCION OPEN CON INPUT #
20 DIM A$(40)
30 PRINT CHR$(125)
40 OPEN #1,4,0,"C:"
50 INPUT #1,A$
60 CLOSE #1
70 FOR I=4 TO 1 STEP -1
80 PRINT "NOMBRE NRO. ";I;" ES ";A$(I*10-9, I*10)
90 NEXT I
100 END
```

PRINT

Formato: PRINT # canal; vars1 [;,] vara1 [;,]
vars2 [;,] etc.
PRINT # canal; "ATARI"

Ejemplos: PRINT # 3; A, B, C, A$, B$, C$
PRINT # 6; "HOLA VENTANA DE TEXTO"

"PRINT #" presenta, muestra o imprime caracteres en un equipo periférico de salida. Su comportamiento es análogo al PRINT sin la opción "canal", excepto que éste último sólo presenta los caracteres en pantalla.

Si el código del periférico de salida es S: o E: la instrucción "PRINT # canal" mostrará los caracteres, también, en pantalla.

Los operadores coma (,) y punto y coma (;), operan de la misma manera que en el PRINT sin la "opción # canal".

Es posible usar el comando PRINT para imprimir datos en pantalla, en cassette (grabadora de cassettes), en diskette (archivo en U. de diskette), su impresora o enviar información hacia las puertas series RS-232. Para ello debe abrirse, previamente, un canal hacia el dispositivo correspondiente, empleando la instrucción OPEN.

```
10 REM USO DE INSTRUCCION OPEN CON PRINT #
20 DIM A$(40),NO$(10)
30 PRINT CHR$(125)
40 A$=" ":A$(40)=A$:A$(2)=A$
50 NO$="          ":REM 10 BLANCOS
60 FOR I=1 TO 4
70 PRINT "INGRESE NOMBRE NRO. ";I
80 INPUT NO$
90 A$(I*10-9,I*10)=NO$
100 NEXT I
110 OPEN #1,8,0,"C:"
120 PRINT #1,A$
130 CLOSE #1
140 END
```

PUT (PU.)

Formato: PUT # canal, expa

Ejemplos:
```
PUT # 1, 65
PUT # CH, NUM
PUT # 4, R
```

La instrucción PUT envía a un periférico un valor numérico único por un canal abierto previamente. El canal debe ser abierto para "salida de datos".

PUT provoca la salida de un valor aritmético (expa), redondeado al entero más próximo. Si el valor no está comprendido entre 0 y 255 entonces es enviado hacia el periférico en la forma llamada "módulo 256", es decir, 256 sería enviado como 0 (256 x 1 + 0), 285 como 29 (256 x 1 + 29), 725 como 213 (256 x 2 + 213), etc.

La instrucción PUT puede enviar información a distintos dispositivos tales como: pantalla (S: o E:), cassette (C:), diskette (D:NOMBARCH), impresora (P:), puertas series (R[n]:). Previamente, por supuesto, deberá abrirse el canal correspondiente (con OPEN) hacia el periférico respectivo.

```
10 REM GRABACION CON INSTRUCCION PUT
20 PRINT CHR$(125)
30 DIM NO$(10)
40 NO$="          ":REM 10 BLANCOS
50 POSITION 2,5:PRINT "INGRESE SU NOMBRE (10
   CARACTERES) ";:INPUT NO$
60 OPEN #1,8,0,"C:"
70 FOR I=1 TO LEN(NO$)
80 PUT #1,ASC(NO$(I,I))
90 NEXT I
100 PUT #1, 155
120 CLOSE #1
130 END
```

GET (GE.)

Formato: GET # canal, vara

Ejemplos: GET # 1, A
GET # CH, AUTO

La instrucción GET recupera un valor numérico único desde un periférico determinado a través de un canal abierto previamente.

El canal debe ser abierto para "entrada" de datos. GET asigna el valor numérico (comprendido entre 0 y 255) que está leyendo desde el periférico a una variable numérica (vara).

La instrucción GET puede recibir información desde distintos periféricos, tales como: teclado (K:), grabadora de cassette (C:), Unidad de Disco (D:), pantalla (S: o E:), puertas series (R[n]:). Previamente deberá abrirse el canal correspondiente (con OPEN) hacia el periférico respectivo.

```
10 REM USO DE INSTRUCCION OPEN CON GET
20 REM LECTURA DEL DIRECTORIO DEL DISCO
30 DIM A$(25)
40 PRINT CHR$(125)
50 TRAP 100
60 OPEN #1,6,0,"D:*.*"
70 INPUT #1,A$
80 PRINT A$
90 GOTO 70
100 CLOSE #1
110 END
```

```
10 REM USO INSTRUCCION OPEN CON GET
20 PRINT CHR$(125)
30 POSITION 17,3:PRINT "MENU"
40 POSITION 5,6:PRINT "1.- MICHAEL JACKSON"
50 POSITION 5,8:PRINT "2.- ENRICO CARUSSO"
60 POSITION 5,10:PRINT "3.- THE BEATLES"
70 POSITION 5,12:PRINT "4.- MIGUEL BOSE"
80 OPEN #1,4,0,"K:"
 90 POSITION 5,16:? "ELIJA UNA OPCION (1-4) ";: GET #1,A
100 CLOSE #1
110 ON VAL(CHR$(A)) GOSUB 1000,2000,3000,4000
120 END
1000 POSITION 5,18;PRINT "CANTANTE DE COLOR AMERICANO"
1010 POSITION 5,19:PRINT "FAMOSO POR SU ALBUM THRILLER"
1020 RETURN
2000 POSITION 5,18:PRINT "CANTANTE DE OPERA ITALIANO"
2010 POSITION 5,19:PRINT "FALLECIDO EN 1921"
2020 RETURN
3000 POSITION 5,18:PRINT "GRUPO ROCK INGLES DE LA
     DECADA"
3010 POSITION 5,19:PRINT "DEL 60 CREADORES DE  TODA  UNA
     MODA"
3020 RETURN
4000 POSITION 5,18:PRINT "DESTACADO CANTANTE ESPAÑOL"
4010 POSITION 5,19:PRINT "DE GRAN LLEGADA EN LA
     JUVENTUD"
4020 RETURN
```

```
10 REM LECTURA CON INSTRUCCION GET
20 PRINT CHR$(125)
30 OPEN #1,4,0,"C:"
40 GET #1,A
50 IF A=155 THEN CLOSE #1:END
60 ? CHR$(A);
70 GOTO 40
```

NOTE (NO.)

Formato: NOTE # canal, varsect, varbyte

Ejemplos: NOTE #5, SECTOR, BYTE
NO. #F2, 5, B

La instrucción NOTE determina o indica la ubicación del puntero de archivo correspondiente al archivo especificado en la instrucción OPEN, precedente.

Esta instrucción asigna a la variable varsect el número del sector y a la variable varbyte el número del byte dentro del sector, en que se halla el archivo especificado. Tales variables no pueden ser elementos de un arreglo (array).

El canal que se debe abrir previamente puede abrirse para cualquier operación o tarea.

POINT (P.)

Formato: POINT # canal, varsect, varbyte

Ejemplos: POINT # 5, SECTOR, BYTE
P. #f2, S, B

La instrucción POINT lleva el puntero de archivo al sector del diskette especificado por el valor de la variable numérica varsect y al byte, dentro de ese sector, especificado por la variable numérica varbyte. Si varsect está fuera de los límites del archivo se producirá un error. El valor de varbyte debe estar comprendido entre 0 y 125.

El canal que se debe abrir previamente, hacia el archivo en cuestión, debe serlo para una operación de entrada, actualización (update) o de agregar (append). Vea OPEN.

Ejemplos del uso de NOTE y POINT se encuentran en la sección 6.10.2. Ejemplo de Archivo Indexado.

STATUS (ST.)

Formato: STATUS # canal, vara

Ejemplos: STATUS # 3, EST
ST. # K, S

Esta instrucción asigna a la variable el "código status", de la última actividad desarrollada por el canal indicado. Si el código es mayor que 128 indicará un STATUS correspondiente a un ERROR.

Si la variable vara es un elemento de un arreglo, el BASIC no indica error al ejecutar la instrucción STATUS. Sin embargo, referencias posteriores al mismo arreglo provocarán errores, hasta que el arreglo sea redimensionado.

```
10 REM INSTRUCCION STATUS
20 PRINT CHR$(125)
25 REM INGRESE UN NOMBRE DE PROGRAMA
30 OPEN #1,4,0,"D:XXXXXXXX.YYY"
40 TRAP 500
50 GET #1,A
60 PRINT CHR$(A):
70 STATUS #1,COD
80 PRINT "ESTADO DE LA OPERACION ";COD
90 GOTO 50
100 END
500 REM ERROR
510 STATUS #1,COD
520 PRINT "ERROR ";COD
530 PRINT "EN LA LINEA ";PEEK(186)+PEEK(187)*256
540 END
```

Los códigos de información de la instrucción STATUS son:

Código decimal	Significado
1	Operación terminada y OK
3	Fin de archivo está próximo: la siguiente lectura recupera la última información del archivo (*)
128	Operación de ENTRADA o SALIDA interrumpida con BREAK
129	Canal abierto previamente
130	Aparato Inexistente
131	Intento de Entrada por un canal de Salida
132	Error de Sintaxis en un comando X1O
133	Canal no Abierto previamente
134	Canal de Salida Desconocido
135	Intento de Salida por un canal de Entrada
136	Fin de archivo
137	Fichero truncado
138	Aparato No Responde
139	Aparato no funciona o no puede ejecutar la instrucción
140	Error en la comunicación entre el aparato y el computador
141	Cursor fuera de pantalla
142	Inconsistencia en la comunicación aparato computador
143	Error en la grabación o lectura desde cassette o diskette
144	Intento de escritura en un diskette protegido
145	Error de Modo de gráfico (GRAPHICS) o verificación
146	Función no implementada
147	Memoria insuficiente para modo gráfico
160	Número de Unidad de Disco desconocido
161	Demasiados archivos abiertos
162	Diskette lleno
163	Error irrecuperable del sistema
164	Número de archivo no corresponde
165	Error en el nombre del archivo
166	Error en la instrucción POINT
167	Archivo protegido
168	Comando XIO desconocido
169	Directorio del diskette lleno
170	Archivo inexistente
171	Instrucción POINT no válida
172	Intento de añadir datos a un archivo con DOS distinto del original
173	Sectores malos al momento de formatear

* Este código difiere del código de error del sistema operativo.

XIO (X_)

Formato: XIO cmd, # canal, expa1, expa2, disp.

Ejemplos: XIO 18, #6, 0, 0, "S:"
XIO BLOQ, #3, 0, 0, MARCH1$

XIO es una instrucción general de entrada/salida. Puede ejecutar una amplia variedad de operaciones de entrada y de salida. El valor de cmd especifica la operación. La tabla 4-4 muestra las posibilidades.

El canal (# canal) debe estar abierto para entrada o salida según corresponda, excepto para la instrucción XIO 3 (OPEN). El parámetro final, disp, selecciona el dispositivo o periférico de entrada o salida. Los parámetros restantes proporcionan información adicional; su uso exacto depende de la operación de que se trate. Todos los parámetros deben estar siempre presentes, aunque no todos son siempre usados.

```
10 REM INSTRUCCION XIO
20 GRAPHICS 7
30 COLOR 3:POKE 765,3
40 PLOT 80,45:DRAWTO 80,35:DRAWTO 45,35
50 POSITION 45,45:XIO 18,#1,0,0,"S:"
60 COLOR 2
70 POKE 765,2
80 PLOT 45,45:DRAWTO 45,35:DRAWTO 35,35
90 POSITION 35,45:XIO 18,#4,0,0,"S:'
100 FOR Q=0 TO 1000:NEXT Q
110 COLOR 1
120 POKE 765,1
130 PLOT 80,50:DRAWTO 80,30:DRAWTO 30,30
140 POSITION 30,50:XIO 18,#6,0,0,"S:'
150 END
```

TABLA 4-4
COMANDOS XIO

Acción	Comando	Equiv. BASIC	expa1	expa2
General:				
Abrir canal	3	OPEN	Tabla 4-1	Tabla 4-2
Leer línea	5	INPUT	0	0
Sacar (Get) caracter	7	GET	0	0
Escribir línea	9	PRINT	0	0
Poner (Put) caracter	11	PUT	0	0
Cerrar canal	12	CLOSE	0	0
Status del canal	13	STATUS	0	0
Pantalla de gráficos:				
Dibujar línea (1)	17	DRAWTO	0	0
Pintar (llenar) área (2)	18	Ninguno	0	0
Diskette: (3)				
Renombrar archivo (4)	32	Menú DOS	0	0
Borrar archivo (5)	33	Menú DOS	0	0
Proteger archivo (5)	35	Menú DOS	0	0
Desproteger archivo (5)	36	Menú DOS	0	0
Mover puntero (5, 6)	37	POINT (6)	0	0
Encontrar puntero de archivo (5,6)	38	NOTE (6)	0	0
Formatear diskette (5)	254	Menú DOS	0	0

NOTAS:

(1) Mueve el cursor al comienzo de línea con instrucción POSITION antes de XIO 17.

(2) Use POKE 765, expa para elegir registro de color y trace límites verticales antes de XIO 18.

(3) DOS debe estar en memoria.

(4) El parámetro "disp" de XIO 32 especifica el archivo cuyo nombre cambiará.

(5) El parámetro "disp" especifica el archivo.

(6) No implementada en versión DOS 1.0

El 4to. parámetro de la instrucción OPEN, espec perif, señala el equipo periférico al cual el canal estará asociado. La tabla 4-3, a continuación, indica los nombres standarizados para los periféricos.

Periférico	Nombre
Cassettera	C:
U. Diskette	D[n]:NOMBARCH.EXT
Editor de pantalla	E:
Teclado	K:
Impresora	P:
RS-232 puerta serie	R[n]:
Pantalla	S:

TABLA 4-3

Nombres de periféricos en OPEN

5.- FUNCIONES ARITMETICAS, TRIGONOMETRICAS Y DE USO ESPECIAL

ABS

Formato: ABS (expa)
Ejemplo: IF A=ABS (A) THEN PR. "POSITIVO"

ABS es una función que entrega el valor absoluto de la expresión aritmética ó numérica sobre la que se aplica.

```
10 REM FUNCION ABS
20 PRINT CHR$(125)
30 FOR I=1 TO 10
40 A=INT(RND(0)*-100)+50
50 PRINT A:
60 IF A=ABS(A) THEN PRINT "[ESC][CLR SET TAB]
   POSITIVO":GOTO 80
65 REM ESCRIBA EL MENSAJE EN VIDEO INVERSO
70 PRINT "[ESC][CLR SET TAB]NEGATIVO"
80 NEXT I
90 END
```

CLOG

Formato : CLOG (expa)
Ejemplo : A = B * CLOG (A)

CLOG entrega el logaritmo común o decimal de un número. Si expa < 0 se producirá un error.

```
10 REM FUNCION CLOG
20 PRINT CHR$(125)
30 POSITION 2.5:PRINT "INGRESE UN NUMERO : ":: INPUT A
40 IF A>0 THEN 120
50 FOR I=1 TO 10
60 POSITION 2.5:PRINT "ERROR EN INGRESO DE VALOR
   NUMERICO"
70 FOR C=1 TO 50:NEXT C
75 REM ESCRIBIR EL MENSAJE EN VIDEO INVERSO
80 POSITION 2.5:PRINT "ERROR EN INGRESO DE VALOR
   NUMERICO
90 FOR C=1 TO 50:NEXT C
100 NEXT I
110 GOTO 20
120 POSITION 2,7:PRINT "EL LOGARITMO DECIMAL DE ";A
130 POSITION 2,9:PRINT "ES ";CLOG(A)
140 POSITION 2,11:PRINT "O 10 ^ ";CLOG(A);" ES ";A
150 END
```

EXP

Formato: EXP (expa)
Ejemplo: K=EXP (J)

EXP calcula la expresión e elevada a la potencia expa, en que e es la base de los logaritmos naturales (e=2.71828179).

```
10 REM FUNCION EXP
20 PRINT CHR$(125)
30 GRAPHICS 8:COLOR 1:PLOT 159,79
40 PLOT 5,79:DRAWTO 314,79
50 PLOT 159,5:DRAWTO 159,155
60 FOR X=0 TO 4 STEP 0.1
70 Y=EXP(X)
80 PRINT "LA FUNCION EXP(";X;") VALE ";Y
90 FOR I=1 TO 500:NEXT I
100 DRAWTO 159+X,79-Y
110 NEXT X
```

INT

Formato: INT (expa)
Ejemplo: N= (INT (N/2)*100+0.5)/100

Devuelve la parte entera de un número. Entrega el mayor entero menor o igual que el valor de expa.

```
10 REM FUNCION INT "REDONDEO"
20 DIM PE$(3),PD$(7)
30 PRINT CHR$(125)
40 POSITION 2,5:PRINT "INGRESE UN NUMERO DECIMAL"
   ;:INPUT NU
50 PE$=STR$(INT(NU))
60 PD$=STR$(NU-INT(NU))
70 PRINT "LA PARTE ENTERA ES ";PE$
80 PRINT "LA PARTE DECIMAL ES ";PD$
90 IF VAL (PD$)>0.5 THEN 120
100 PRINT "REDONDEANDO ES ";VAL(PE$)
110 GOTO 130
120 PRINT "REDONDEANDO ES ";VAL(PE$)+1
130 END
```

LOG

Formato: LOG (expa)
Ejemplo: R=P*LOG(Q)

LOG calcula el logaritmo natural (base e) de expa. Si expa < 0 se producirá un ERROR.

```
10 REM FUNCION LOG(X)
20 GRAPHICS 8:COLOR 1:PLOT 159,79
30 PLOT 5,79:DRAWTO 314,79
40 PLOT 159,5:DRAWTO 159,154
50 FOR X=0.1 TO 100
60 Y=LOG(X)
70 DRAWTO 159+X,79-Y
80 NEXT X
90 PRINT "FUNCION Y=LOG(X)"
100 END
```

RND

Formato: RND (expa)
Ejemplo: V=RND(0)*8

Genera un número en coma flotante mayor o igual que 0 y menor que 1. El valor de expa no tiene influencia sobre el valor del número aleatorio (random), pero debe estar presente.

```
10 REM FUNCION INT GENERADORA
20 PRINT CHR$(125)
30 POSITION 2,4:PRINT "DESEA GENERAR NUMEROS ENTRE 1
   Y ";:INPUT N
40 POSITION 2,8:PRINT "CUANTOS ";:INPUT I
50 FOR A=1 TO I
60 PRINT "GENERO EL ";INT(RND(0)*N)+1
70 NEXT A
80 END
```

SGN

Formato: SGN (expa)
Ejemplo: IF SGN (J)=-1 THEN PR. "NEGATIVO"

La función SGN identifica un número como positivo, negativo o cero. Entrega +1 si expa es positivo, -1 si es negativo y 0 si es cero.

SQR

Formato: SQR
Ejemplo: HIPOY=SQR (CAT1^2 + CAT2^2)

La función SQR calcula la raíz cuadrada de un número positivo. Un valor negativo de expa, causará un ERROR. SQR (expa) opera más rápido que (expa) ^ 0.5.

```
10 REM FUNCION SQR(X)
20 GRAPHICS 8
30 COLOR 1
40 PLOT 5,79:DRAWTO 314,79
50 PLOT 159,5:DRAWTO 159,154
60 PLOT 159,79
70 FOR X=0 TO 100
80 Y=SQR(X)
90 DRAWTO 159+X,79-Y
100 NEXT X
110 PRINT "FUNCION Y=SQR(X)
```

ATN

Formato: ATN (expa)
Ejemplo: PRINT ATN (ANG)

Entrega el arcotangente del argumento, expa. Calcula el arcotangente de expa en radianes o en grados si la instrucción DEG está explicitada. El valor calculado está comprendido en el rango - /2 a + /2.

COS

Formato: COS (expa)
Ejemplo: COS (3.1415)

Calcula el coseno de un ángulo. Calcula el coseno de expa radianes o expa grados si la instrucción DEG está explicitada.

SIN

Formato: SIN (expa)
Ejemplo: SIN (BETA)

Calcula el seno de un ángulo. Calcula el seno de expa radianes o expa grados si la instrucción DEG está explicitada.

DEG

Formato: DEG
Ejemplos: DEG
DE.

La función DEG indica el BASIC que los argumentos de las funciones trigonométricas están expresados en grados y no en radianes. Después de ejecutar la instrucción DEG, el BASIC considera los argumentos de las funciones trigonométricas como grados. Para volver a radianes use la instrucción RAD o apague el computador y vuelva a prenderlo o presione la tecla SYSTEM RESET. El BASIC vuelve a radianes, también, después de una instrucción NEW o RUN.

RAD

Formato: RAD
Ejemplo: RAD

La función RAD indica al BASIC que los argumentos de las funciones trigonométricas están expresadas en radianes y no en grados. Después de ejecutar la instrucción RAD, el BASIC considera los argumentos de las funciones trigonométricas como radianes. Para pasar a grados use la instrucción DEG. El BASIC se va a radianes cuando ejecuta una instrucción NEW o RUN, cuando se presiona la tecla SYSTEM RESET o cuando se apaga el computador y se vuelve a prender.

Este es el modo normal de operación.

```
10 REM FUNCIONES TRIGONOMETRICAS
20 DIM MEN$(12)
30 DEG
40 PRINT CHR$(125)
50 POSITION 7,4:PRINT "FUNCIONES TRIGONOMETRICAS"
60 TRAP 60:POSITION 2,6:PRINT "INGRESE UN ANGULO ENTRE
   0 Y 360 :";: INPUT ANG
70 POSITION 2,8:PRINT "EL COSENO DE ";ANG;" ES ";
   COS(ANG)
80 POSITION 2,10:PRINT "EL SENO DE ";ANG;" ES ";
   SIN(ANG)
90 LIN=110:TRAP 2000
100 MEN$=STR$(SIN(ANG)/COS(ANG))
110 POSITION 2,12:PRINT "LA TANGENTE DE ":ANG;"
    ES";MEN$
120 LIN=140:TRAP 2000
130 MEN$=STR$(COS(ANG)/SIN(ANG))
140 POSITION 2,14:PRINT "LA CO-TANGENTE DE ";ANG;" ES
";MEN$
150 LIN=170:TRAP 2000
160 MEN$=STR$(1/COS(ANG))
170 POSITION 2,16:PRINT "LA SECANTE DE ";ANG;" ES";MEN$
180 LIN=200:TRAP 2000
190 MEN$=STR$(1/SIN(ANG))
200 POSITION 2,18:PRINT "LA CO-SECANTE DE ";ANG;" ES "
    ;MEN$
210 END
2000 MEN$="VALOR NULO":REM MENSAJE EN VIDEO INVERSO
2010 TRAP 40000
2020 GOTO LIN
```

ADR

Formato: ADR (string)
Ejemplo: A= USR (ADR,(SUBR$))

La función ADR aplicada a una variable o constante alfanumérica (string) entrega el valor decimal de la dirección de partida de almacenamiento del string en la memoria del computador.

Es posible poner un programa en lenguaje de máquina en una variable alfanumérica del lenguaje BASIC. La función ADR puede determinar la dirección de partida de la variable string, que es la misma que la dirección de partida del programa en lenguaje de máquina.

```
10 DIM A$(30)
20 RESTORE 260
30 FOR I=1 TO 28
40 READ A
50 A$(I,I)=CHR$(A)
60 NEXT I
70 GRAPHICS 10
80 RESTORE 270:FOR I=1 TO 8:READ A:POKE 704+1,A+48:
   NEXT I
90 Q=1
100 FOR Y=0 TO 55 STEP 2
110 FOR X=0 TO 8
120 COLOR Q:IF Q>15 THEN COLOR 31-Q
130 FOR J=0 TO 60 STEP 12
140 PLOT X+J,Y:PLOT X+J, Y+1
150 PLOT X+J, Y+60:PLOT X+J, Y+61
160 PLOT X+J, Y+120:PLOT X+J,Y+121
170 NEXT J
180 Q=Q+1:IF Q>8 THEN Q=1
190 NEXT X
200 NEXT Y
210 FOR T=1 TO 200
220 X=USR (ADR(A$))
230 FOR J=1 TO 18:NEXT J
240 NEXT T
250 GOTO 210
260 DATA 104, 162, 0, 172, 193, 2,  189,  194,  2,  157,
193, 2, 232
265 DATA 224, 8, 144, 245, 140, 200, 2, 96, 65, 65,  65,
65, 65, 65
270 DATA 2, 4, 6, 8, 6, 4, 2, 2
```

FRE

Formato: FRE (expa)
Ejemplo: ? FRE (0)

Esta función permite saber cuantos bytes de memoria quedan disponibles en un determinado momento. La memoria disponible es aquella que no ha sido ocupada por el sistema operativo el DOS, el sistema de manejo de pantalla o un programa en BASIC y sus datos.

El valor de expa puede ser cualquiera, pero ocurrirá un error si queda fuera del rango +9.99999999E+98.

PEEK

Formato: PEEK (dirmem)
Ejemplo: LMARG=PEEK (82)

Entrega el contenido de una dirección de memoria. El valor entregado es el equivalente decimal del caracter guardado en esa ubicación de memoria.

POKE (POK.)

Formato: POKE dirmem, expa
Ejemplo: POKE 756,204

El parámetro expa debe tener un valor entero comprendido entre 0 y 255. Este valor es escrito en memoria en la ubicación indicada (dirmem).

Use POKE con precaución, si tiene un programa en memoria. Algunas ubicaciones de memoria contienen información necesaria para la operación del computador. Variar los valores de las ubicaciones de memoria puede alterar su programa o "colgar" el sistema.

El cero de esta instrucción permite lograr una gran cantidad de efectos especiales en sus programas. Ensaye algunos de los valores que entregamos a continuación.

```
10 REM COMANDOS PEEK Y POKE
20 REM LISTA DE VARIABLES
30 DIM VARNOM$(1),NOMBRE$(1)
40 REM LEE EL INICIO DE LA LISTA DE VARIABLES
50 DIR=PEEK(130)+PEEK(131)*256
60 PRINT CHR$(125)
770 PRINT "LISTA DE LAS VARIABLES"
80 REM LEE UN CARACTER Y VERIFICA SI NO ES FIN
90 A=PEKK(DIR):IF A=0 THEN END
100 REM IMPRIME EL CARACTER
110 IF A<128 THEN PRINT CHR$(A);:GOTO 140
120 PRINT CHR$(A-128)
130 REM PROXIMO CARACTER Y REPITE PROCESO
140 DIR=DIR+1:GOTO 90
```

```
10 REM COMANDOS PEEK Y POKE
20 REM PROGRAMA QUE IMPRIME EL
30 REM NUMERO DE LINEA Y SU
40 REM DIRECCION EN LA MEMORIA
45 PRINT CHR$(125)
50 REM LEE LA DIRECCION DE LA
60 REM PRIMERA LINEA DE PROGRAMA
70 DIR=PEEK(136)+PEEK(137)*256
80 REM LEE EL NUMERO DE LINEA
90 LNUM=PEEK(DIR)+PEEK(DIR+1)*256
100 REM VERIFICA FIN DE PROGRAMA
110 IF LNUM=32768 THEN END
120 REM IMPRIME LINEA Y DIRECCION
130 PRINT "LINEA #";LNUM;"DIREC. DE INICIO ";DIR
140 REM LEE LA DIRECCION PROXIMA LINEA
150 DIR=DIR+PEEK(DIR+2)
160 GOTO 90
```

```
10 REM INSTRUCCION POKE
20 REM TABULACION DE MARGENES
30 REM POKE 82,X MARGEN IZQUIERDO
40 REM POKE 83,X MARGEN DERECHO
50 PRINT CHR$(125)
60 POKE 82,10:POKE 83,29
70 FOR I=1 TO 40
80 PRINT "ATARI";
90 PRINT " "
100 FOR A=1 TO 50:NEXT A
110 NEXT I
120 PRINT CHR$(125)
130 LIST
140 FOR R=1 TO 2000:NEXT R
150 RUN
```

PADDLE

Formato: PADDLE (expa)
Ejemplo: PLOT PADDLE (0)/6, PADDLE (1)/12

Entrega un valor entero entre 1 y 228 dependiendo de la rotación del paddle correspondiente o de la resistencia eléctrica de un dispositivo conectado al enchufe controlador de juego. Los paddles están numerados de 0 a 7. Si el número del paddle es menor que 0 o mayor que 255 se producirá un ERROR. Si el número del paddle está comprendido entre 8 y 255, PADDLE entrega un número que es impredecible.

PTRIG

Formato: PTRIG (expa)
Ejemplo: IF PTRIG (1)=0 THEN PRINT "BRAVO !!!"

Determina si el botón del paddle especificado está gatillado o no. Si el botón del paddle número expa está presionando el valor entregado es 0. Si no lo está es 1. Los paddles están numerados de 0 a 7. Si el número del paddle es menor que 0 o mayor que 255, se producirá un ERROR. Si el número del paddle está comprendido entre 8 y 255, PTRIG entrega un número impredecible.

STICK

Formato: STICK (expa)
Ejemplo: IF STICK (0) = 14 THEN J=J+1

Entrega un número entero comprendido entre 0 y 15, dependiendo de la posición del joystick correspondiente. Los joystick están numerados 0 a 3. Si el número del joystick es menor que 0 o mayor que 255, se producirá un ERROR. Si el número del joystick es entre 0 y 255, STICK entrega un número impredecible.

AL CENTRO	15
ADELANTE	14
ADELANTE-DERECHA	6
DERECHA	7
ATRAS-DERECHA	5
ATRAS	13
ATRAS-IZQUIERDA	9
IZQUIERDA	11
ADELANTE-IZQUIERDA	10

VALORES DE STICK
Según la posición del Joystick

STRIG

Formato: STRIG (expa)
Ejemplo: IF STRIG (1)=0 THEN PRINT "BRAVO !!!"

Determina si el botón del joystick correspondiente está gatillado o no. Si el botón del joystick está presionado el número entregado es 0. Si no lo está es 1. Los joysticks están numerados de 0 a 3. Si el número del joystick es menor que 0 o mayor que 255, ocurrirá un ERROR. Si el número del joystick está entre 4 y 255, STRIG entrega un número impredecible.

```
10      PROGRAMA PARA INSTRUCCION
20      STICK Y STRIG
30 REM LEE SEIS PREGUNTAS
40 REM DE CUATRO ALTERNATIVAS
50 REM T1 = Nro. DE PREGUNTA A LEER ES AL AZAR
60 REM T2 = Nro. DE RESPUESTA A LEER ES AL AZAR
70 REM PR$ = STRING DONDE SE GUARDA LA PREGUNTA LEIDA
80 REM R1$ = STRING DONDE SE GUARDA LA PRIMERA RESPUESTA LEIDA (CORRECTA)
90 REM R2$ = STRING DONDE SE GUARDA LA SEGUNDA RESPUESTA LEIDA (INCORRECTA)
100 REM R3$ = STRING DONDE SE GUARDA LA TERCERA RESPUESTA LEIDA (INCORRECTA)
110 REM R4$ = STRING DONDE SE GUARDA LA CUARTA RESPUESTA LEIDA (INCORRECTA)
120 DIM T1(6),T2(4),PR$(40),R1$(20),R2$(20),R3$(20),R4$(20),R$(20)
130 GRAPHICS 0
140 POKE 712,170:POKE 710,170:POKE 709,4:POKE 752,0:POKE 756,204
150 OPEN #1,4,0,"K"
160 POSITION 5,10:? " Posee Joystick (S/N) : ";:GET #1,A
170 IF A<>83 AND A<>78 THEN 160
180 CLOSE #1
190 IF A<>83 THEN END
200 FOR I=1 TO 6
210 T1(I)=INT((RND(0)*6)+1)
220 IF I=1 THEN GOTO 270
230 A=I-1
240 FOR K=1 TO A
250 IF T1(K)=T1(I) THEN GOTO 210
260 NEXT K
270 NEXT I
280 TPRE=6:RCO=0:RIN=0
290 FOR P=1 TO 6
300 FOR I=1 TO 4
310 T2(I)=INT((RND(0)*4)+1)
320 IF I=1 THEN GOTO 370
330 A=I-1
340 FOR K=1 TO A
350 IF T2(K)=T2(I) THEN GOTO 310
360 NEXT K
370 NEXT I
380 GRAPHICS 0:POKE 82,7:POKE 756,204:POKE 710,170:POKE 712,170
385 POKE 709,3:POKE 752,0
390 POSITION 1,2:? "   Seleccione la respuesta correcta   "
400 POSITION 1,3:? "      moviendose con el Joystick      "
410 POSITION 1,4:? "   y presionando el boton rojo para   "
420 POSITION 1,5:? "        confirmar su eleccion         "
430 RESTORE (890+T1(P)*10)+((890+T1(P)*10)-900)
440 READ PR$,R1$,R2$,R3$,R4$
450 POSITION 3,7:? P;")";" ";PR$
460 FOR R=1 TO 4
470 IF T2(R)=1 THEN R$=R1$:RES=8+R
480 IF T2(R)=2 THEN R$=R2$
490 IF T2(R)=3 THEN R$=R3$
500 IF T2(R)=4 THEN R$=R4$
```

```
510 POSITION 7,8+R:? CHR$(96+R);")";" ";R$
520 NEXT R
530 POSITION 7,7
540 ? CHR$(29)
550 SC=7:SR=7
560 LC=7:RC=7:TR=7:BR=10
570 DLY1=30
580 GOSUB 660
590 IF SW=1 THEN GOTO 830
600 FOR X=1 TO 10
610 POSITION 14,20:? "  CORRECTO  ":FOR I=1 TO 5:NEXT I
620 POSITION 14,20:? "            ":FOR I=1 TO 5:NEXT I
630 SOUND 3,R,I,14
640 NEXT X
650 SOUND 3,0,0,0:RCO=RCO+1:POKE 752,0:RES=0:GOTO 830
660 IF STRIG(0)=0 THEN GOTO 730
670 FOR J=1 TO DLY1:NEXT J
680 IF STICK(0)=7 AND SC<>RC THEN SC=SC+1:? CHR$(31)
690 IF STICK(0)=11 AND SC<>LC THEN SC=SC-1:? CHR$(30);
700 IF STICK(0)=13 AND SR<>BR THEN SR=SR+1:? CHR$(29);
710 IF STICK(0)=14 AND SR<>TR THEN SR=SR-1:? CHR$(28);
720 GOTO 660
730 POKE 752,1:IF PEEK(84)=RES THEN SW=0:RES=0:RETURN
740 FOR X=1 TO 10
750 POSITION 14,20:? " INCORRECTO ":FOR I=1 TO 5:NEXT I
760 POSITION 14,20:? "            ":FOR I=1 TO 5:NEXT I
770 SOUND 2,I,X,14
780 NEXT X
790 POKE 752,0:SOUND 2,0,0,0:RIN=RIN+1
800 POSITION 3,20:? "La respuesta correcta es ":POSITION 7,22:? R1$
810 FOR I=1 TO 300:NEXT I
820 SW=1:RETURN
830 NEXT P
840 REM PUNTAJE
850 ? CHR$(125):POKE 752,1
860 POSITION 5,10:? "Respuestas correctas   : ";RCO
870 POSITION 5,12:? "Respuestas incorrectas : ";RIN
880 POSITION 10,14:? "Puntaje  : ";INT((RCO*100)/TPRE);" %"
890 POKE 82,2:POKE 752,0:FOR R=1 TO 1000:NEXT R:POKE 752,2:END
895 REM PREGUNTAS Y RESPUESTAS                    NUMERADAS DE 10 EN 10
900 DATA 1ro en cruzar el C.de la Mancha,Bleriot(Bleriot XI)
905 DATA Dumont(Wright IV)
910 DATA A.V.Roe(Avro 104),Hensen(Spad 13)
920 DATA Von Richthofen uso el avion,Fokker DR-1 triplano,Taube biplano
930 DATA Albatros biplano,Nieuoort 17 caza
940 DATA Avion aleman usado en el Artico,Fokker F-7 trimotor,Junkers bimotor
950 DATA Gotha bimotor,Ryan monomotor
960 DATA El Spirit of st.Louis cruzo el,Oceano Atlantico,Oceano Indico
970 DATA Oceano Pacifico,Oceano Artico
980 DATA 1er avion seguro-veloz-economico,Douglas DC-3 ,Douglas DC-4
990 DATA Boeing 247,Boeing 747
1000 DATA El fortaleza volante se llamaba,Boeing B-17A,Boeing B-25A
1010 DATA Boeing B29-B,Boeing B-37B
```

USR

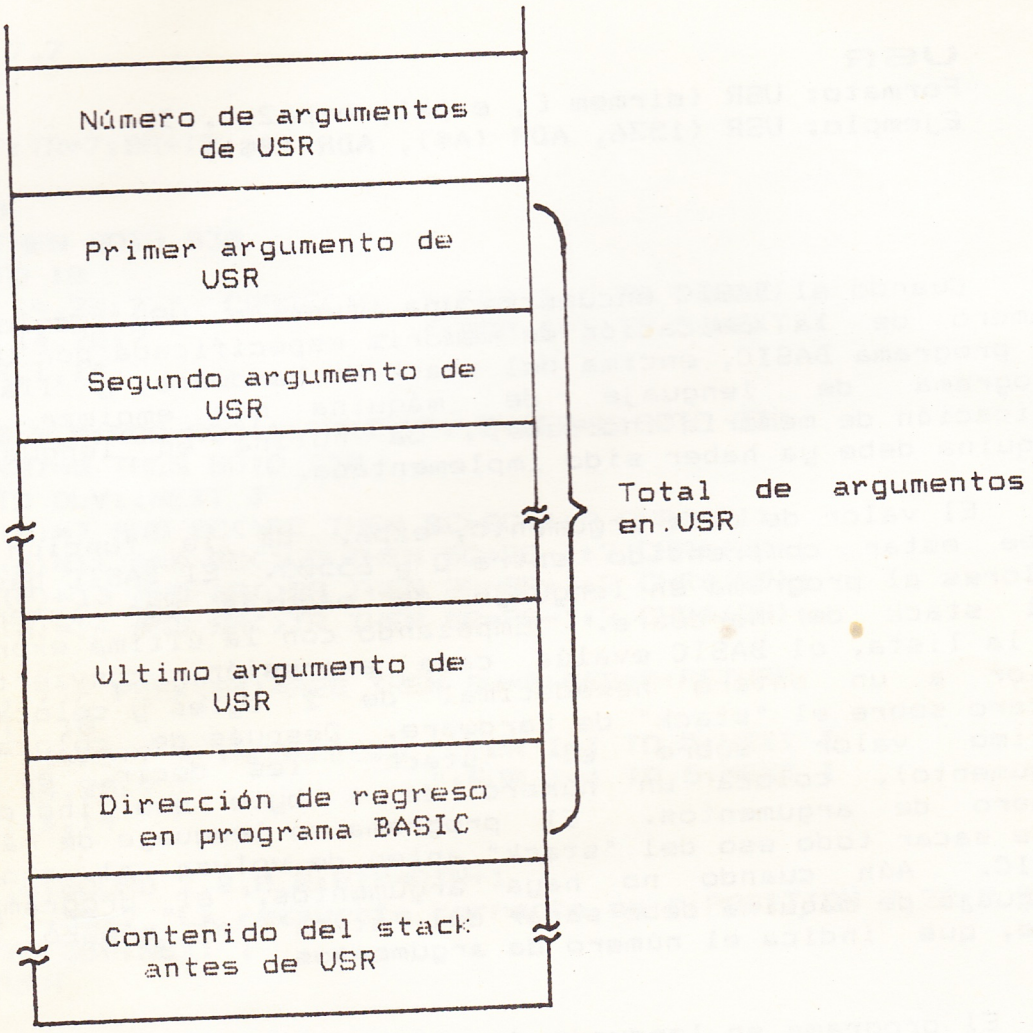
Formato: USR (dirmem [, expa1, expa2....])
Ejemplo: USR (1536, ADR (A$), ADR (B$))

Cuando el BASIC encuentra una función USR, coloca el número de la ubicación de memoria especificada por USR, en el programa BASIC, encima del stack de hardware y llama al programa de lenguaje de máquina que empieza en la ubicación de memoria indicada. La rutina en lenguaje de máquina debe ya haber sido implementada.

El valor de cada argumento, expa, de la función USR, debe estar comprendido entre 0 y 65535. El BASIC pasa los valores al programa en lenguaje de máquina por intermedio del stack de hardware. Empezando con la última expresión de la lista, el BASIC evalúa cada expresión, convierte el valor a un entero hexadecimal de 2 bytes y coloca este entero sobre el "stack" de hardware. Después de colocar el último valor sobre el "stack" (es decir, el 1er. argumento), coloca un número de 1 byte que indica el número de argumentos. El programa en lenguaje de máquina debe sacar todo eso del "stack" antes de volver al programa BASIC. Aún cuando no haya argumentos, el programa en lenguaje de máquina debe sacar del "stack" el número de 1 byte, que indica el número de argumentos.

El programa en lenguaje de máquina vuelve al programa en BASIC ejecutando una instrucción RTS de lenguaje assembler, que saca la dirección a donde debe volver del "stack" de hardware.

NOTA: Se entiende como "STACK" un conjunto de registros o direcciones acumulados como en una pila.



STACK DE HARDWARE Y USR

```
10 REM RUTINA DE RAINBOW
20 REM SE AGREGA AL FINAL DE UN GRAFICO
30 FOR I=1664 TO 1673:READ A:POKE I,A:NEXT I
40 DATA 232, 142, 10, 212, 142, 24, 208, 76, 128, 6
50 ? USR (1664)
```

6.- LA UNIDAD DE DISKETTE

La unidad de diskette o unidad de disco es uno de los componentes más importantes de un sistema de computación. Permite acceder en forma muy rápida a gran cantidad de información. La U. de diskette ATARI 1050 permite almacenar alrededor de 130.000 caracteres en un solo diskette. Esto representa el doble del total de información que puede ser mantenida en un computador de 64 K de RAM. Al apagar el computador la información contenida en RAM desaparece, pero la información almacenada en un diskette permanece intacta.

Los diskettes son conocidos también como discos flexibles o "floppy" disks. El tamaño de estos diskettes es variable. Los hay de 8" de diámetro y también de 5 1/4" y de 3 1/2". La U. de diskette ATARI 1050 emplea diskettes de 5 1/4" de diámetro. Estos diskettes en densidad simple pueden almacenar hasta 90 KBytes (92.160 caracteres) y en densidad dual o extendida hasta 130 KBytes (133.120 caracteres).

En un costado del diskette hay una muesca de protección que permite o impide que se pueda grabar información en el diskette. Si la muesca está descubierta puede escribirse o grabarse información en el diskette. Si la muesca está cubierta o protegida con una pequeña etiqueta, entonces el diskette está protegido contra la escritura. No será posible grabar o escribir en él. (Diskettes de 5 1/4").

Algunos diskettes no tienen muesca. Esto significa que están permanentemente protegidos contra grabación o escritura accidental.

6.1.- ARCHIVOS EN DISKETTE

La unidad de diskette almacena la información en archivos (files) de manera similar a la que uno puede guardar información en un mueble de kárdex. Cada diskette equivale a un cajón de kárdex. Cada archivo corresponde a una carpeta colgante en el cajón. Un diskette puede contener muchos archivos o puede no contener ninguno. El número máximo de archivos por diskette es 64; sin embargo si algunos de los archivos son muy largos el diskette puede llenarse con un número de archivos menor que 64. Un archivo en particular puede ser de cualquier largo que físicamente quepa en el diskette.

Un archivo de diskette puede contener un programa en BASIC o en lenguaje de máquina o bien estar constituído por datos. Las técnicas para leer y escribir en cada caso son distintas.

Cada archivo debe tener un nombre para distinguirlo de los demás archivos en el diskette.

Los nombres de archivo pueden tener hasta 8 caracteres que pueden ser letras mayúsculas y números en cualquier combinación pero empezando por una letra. No se admiten espacios en blanco ni caracteres especiales, tales como $, @, #, /, etc., ni ningún signo de puntuación.

El archivo puede tener un sufijo de hasta 3 caracteres que se agrega a los hasta 8 caracteres mencionados. Este sufijo es llamado "extensión del nombre del archivo". La extensión está separada del nombre por el signo "punto" (.). Los caracteres de la extensión pueden ser letras o números.

Así pués un archivo queda especificado por su nombre y su extensión si la tiene Ejemplo: ARCHIVO 1.BAS.

Muchas veces la extensión se usa para designar el tipo de archivo de que se trata. Por ej.: ARCHIVO1.BAS correspondería a un programa hecho en lenguaje BASIC.

La lista a continuación señala a modo de sugerencia las denominaciones más usadas para designar distintos tipos de archivos.

EXTENSION	TIPO DE ARCHIVO
SYS	Archivos del sistema. Archivos que contienen programas como el DOS o intérpretes de lenguaje como Microsoft BASIC.
BAS	Archivos que contienen programas BASIC grabados en formato "tokenizado" (comando CSAVE o SAVE).
LST	Archivos que contienen programas BASIC grabados en formato ATASCII (comando LIST).
ASM	Programas en lenguaje Assembler en forma de texto.
DAT	Archivos de datos. Se usa también DTA.
TXT	Archivos que contienen texto.
BAK	Archivos de respaldo (BACK UP).
TMP	Archivos temporales. Archivos que sólo serán útiles durante un período de tiempo corto.
OBJ	Archivo o programa objeto. Programas en Assembler pasados a lenguaje máquina.

Una pequeña parte de la memoria del diskette (menos de 3 Kb) está reservada para contener el directorio. El directorio guarda el nombre, ubicación y tamaño de cada archivo. Así, cuando se especifica un archivo, para llevarlo a pantalla, por ej.: el computador busca el nombre en el directorio, para ver si está y dónde está.

6.2.- EL SISTEMA OPERATIVO DE DISCO o D.O.S. (DOS).

El Sistema Operativo de Disco (Disk Operating System: DOS) es un programa en lenguaje de máquina que controla el funcionamiento de la Unidad de Diskette.

Los D.O.S. de ATARI de uso más común son:

DOS 2.0S, DOS 3.0 y últimamente el DOS 2.5.

El DOS 2.0S trabaja en densidad simple y formatea el diskette para una capacidad de 88 Kb. (707 sectores libres). Cada sector tiene 128 Kbytes.

El DOS 3.0 trabaja en densidad simple o densidad dual o extendida. En densidad simple hace que el diskette tenga la misma capacidad que con DOS 2.0S, es decir 88 Kb.

En densidad dual o extendida el diskette es capaz de almacenar 127 Kb. (127 Blocks libres). (1024 Kbytes por block)

El DOS 2.5 trabaja en densidad simple o densidad dual o extendida. En densidad simple tiene la misma capacidad que con DOS 2.0S, es decir, 707 sectores libres después de preparado (Formateado). En densidad dual los sectores libres aumentan a 1010.

Los DOS 2.0 y 3.0 no son compatibles, es decir, no es posible leer con el DOS 3.0 programas escritos en DOS 2.0 y vice-versa. El DOS 3.0 permite la conversión de programas desde DOS 2.0 A DOS 3.0, pero lo inverso no es posible.

El DOS 2.5, es compatible con el DOS 2.0. Con una diskettera 1050 y un diskette formateado en DOS 2.5 es posible COPIAR (COPY) en él directamente los archivos en DOS 2.0.

Quienes tengan la Unidad de Disco ATARI 810 podrán usar aquellos programas en DOS 2.5 que hayan sido preparados y escritos en densidad simple con una ATP 1050.

6.3 Partes del DOS

Los programas que constituyen el DOS 2.0S y DOS 2.5 constan de 2 partes. Una parte (DOS.SYS) graba y carga los programas en BASIC y lee y escribe archivos de datos. La otra parte (DUP.SYS, DUP=Disk Utility Package) constituye un conjunto de programas utilitarios que permiten realizar una serie de operaciones de mantención y manipulación de archivos. Estos programas utilitarios permiten, por ej., copiar archivos, duplicar diskettes, borrar archivos, leer y escribir programas en lenguaje máquina, etc.

El archivo DOS.SYS es necesario cada vez que se usa la Unidad de Diskette, pero la otra parte, DUP.SYS es necesaria sólo cuando se pretende utilizar los programas utilitarios. Dejando en un diskette sólo el DOS.SYS, el espacio que habría ocupado el DUP.SYS puede ser ocupado por un programa o archivo de mayor interés para el programador.

El DOS 3.0 consta de las mismas 2 partes, pero en 3 archivos: FMS.SYS, KCP.SYS y KCPOVER.SYS. El FMS.SYS (File Management System) permite la grabación y lectura de archivos tanto de programas como de datos. El FMS.SYS debe estar incluído en el diskette de programa para evitar tener que cargarlo previamente. Puede ser omitido en el diskette de archivo de datos.

El FMS.SYS es el equivalente al DOS.SYS del DOS 2.0S y DOS 2.5.

Además, el FMS.SYS contiene algunas sub-funciones utilitarias: BORRAR ARCHIVO, RENOMBRAR ARCHIVO, PROTEGER ARCHIVO, DESPROTEGER ARCHIVO Y CARGAR (LOAD).

Los archivos KCP.SYS y KCPOVER.SYS contienen el MENU del DOS y las sub-funciones utilitarias no contenidas en el FMS: GRABAR (SAVE), IR A DIRECCION HEX (GOTO HEX ADDRESS), AL CARTRIDGE, COPIAR ARCHIVO, INICIALIZAR DISKETTE, DUPLICAR DISKETTE Y ACCESO A DOS 2.

6.4 Transferencia de programas y datos

El DOS controla el flujo de información entre el computador y la Unidad de Diskette. La transferencia se realiza en grupos o bloques de 128 caracteres. Para ello el computador reserva un espacio de memoria suficiente para contener un bloque de datos (128 bytes). Este espacio se llama "disk buffer".

Cuando el DOS recibe un pedido por más líneas de programas o más datos, trata de satisfacer el pedido acudiendo al "disk buffer" y tomando la información allí almacenada. Cuando el buffer se vacia, el DOS lo llena leyendo otro block de información del diskette.

De manera análoga, aunque en sentido contrario, se hace la transferencia de información desde el computador al diskette. El DOS llena el buffer con 128 bytes y una vez que está lleno, lo envía para que sea escrito en el diskette.

6.5 Uso de la Unidad de Diskette

Para poder operar con la Unidad de Diskette es decir, para que el computador ATARI pueda ejecutar cualquier comando o instrucción referente a los diskettes, es preciso que el DOS esté cargado en la memoria del computador. La mejor manera de hacerlo es usar un diskette que lo contenga y cargarlo en memoria. Cargar el DOS (booting the disk or booting DOS) significa leer una copia del DOS desde un diskette y colocarlo en la memoria.

Para hacerlo se procede de la siguiente manera:

1.- Encienda la Unidad de Diskette

2.- Coloque un diskette que contenga una copia del Sistema Operativo de Disco. Debe contener por lo menos el archivo DOS.SYS (DOS 2.0S y DOS 2.5) o el archivo FMS.SYS (DOS 3.0).

3.- Apague el computador y vuelva a encenderlo. Con esto la Unidad de Diskette empieza a funcionar. Se escucha un ruido ("rrrrrr") sordo mientras se transfiere el DOS a la memoria. Si se aumenta el volumen del TV, podrán escucharse los "pititos" indicadores del proceso de carga en memoria. Al cabo de unos 20 seg. la palabra READY aparecerá en pantalla.

Llegado a este punto caben 2 alternativas:

1.- Iniciar un programa en lenguaje BASIC y después grabarlo en el mismo o en otro diskette o bien, cargar en memoria un programa o archivo que esté contenido en ese o en otro diskette.

2.- Escribir DOS y presionar [RETURN]. Aparecerá en pantalla el MENU del DOS, después de unos segundos durante los cuales se cargan en memoria los archivos DUP.SYS (DOS 2.0S y DOS 2.5), los archivos KCP.SYS y KCPOVER.SYS (DOS 3.0).

En este segundo caso hay que tener en consideración que el archivo DUP.SYS se carga en un área de memoria ocupada por los programas en BASIC. De modo que si se quieren emplear los programas utilitarios del DOS hay que usarlos antes de hacer o cargar un programa en BASIC, o bien, usarlos después de haber creado un archivo llamado MEM.SAV, que permite guardar la información contenida en el área de memoria que será ocupada por el DUP.SYS. Uno de los programas utilitarios del DOS permite crear el archivo MEM.SAV.

6.6.- El archivo MEM.SAV

En general, cada vez que se ejecuta el comando DOS, el computador busca si existe el archivo MEM.SAV. Si existe dicho archivo, el computador guarda en MEM.SAV la parte de programa ubicada en el área de memoria que será "invadida" por los utilitarios del DUP.SYS y transfiere el control al MENU del DOS. Cuando se vuelve al BASIC, el contenido del archivo MEM.SAV es devuelto al área de memoria primitiva y el programa queda intacto.

Los detalles específicos de cómo opera esta función MEM.SAV para cada uno de los DOS en particular, aparecen explicitados en el respectivo Manual del DOS.

6.7.- MENU del DOS - OPCIONES -

Antes de referirnos al MENU del DOS y sus opciones propiamente tal, diremos algo respecto a los caracteres "comodines" (wild cards) que permitirán un manejo, en algunos casos, mas simplificado y rápido de las opciones.

Muchos de los programas utilitarios piden que se ingrese uno o más nombres de archivo. Siempre es posible ingresar explícitamente el nombre de un archivo. En algunos casos, sin embargo, es más fácil usar nombres ambiguos. Por ej.: si uno no recuerda mas que las 3 primeras letras del archivo, el DOS puede buscar y listar los archivos que empiecen con esos 3 caracteres.

El DOS considera el asterisco (*) y el signo de interrogación (?) como "comodines".

El asterisco (*) reemplaza a todo el nombre de un archivo o a su extensión. Por ej.: el nombre *.BAS significa: todos los archivos cuya extensión es BAS. Análogamente CARTA.* indica a todos los archivos cuyo nombre es CARTA, cualesquiera sea su extensión.

Los caracteres a la derecha de un asterisco no son tomados en cuenta. Por ej.: *TERO.SON es lo mismo que *.SON, pero CASA*.DOC equivaldría a todos los archivos que comienzan con CASA y con extensión DOC.

El asterisco trabaja al revés cuando se le usa en la extensión. En este caso los caracteres a la izquierda del asterisco no son tomados en cuenta. Así por ejemplo la extensión .A* es equivalente a .* pero .*A significa todas las extensiones que terminan en A.

El signo de interrogación (?) puede representar un solo caracter. Por ejemplo el nombre CASA?.BAS reemplaza a todos los archivos cuyo nombre de 5 caracteres empieza por CASA y tienen la extensión BAS.

El nombre ????????.SOL es equivalente a *.SOL.

El Menú típico es el del DOS 2.0S, que es muy similar al del DOS 2.5.

Las opciones del DOS 2.0S son:

A. DISK DIRECTORY
B. RUN CARTRIDGE
C. COPY FILE
D. DELETE FILE (S)
E. RENAME FILE
F. LOCK FILE
G. UNLOCK FILE
H. WRITE DOS FILES
I. FORMAT DISK
J. DUPLICATE DISK
K. BINARY SAVE
L. BINARY LOAD
M. RUN AT ADDRESS
N. CREATE MEM.SAV
O. DUPLICATE FILE

Estas opciones permiten manejar la información almacenada en diskettes.

Al aparecer el MENU del DOS 2.0S, puede observarse debajo del listado de opciones un mensaje que dice:

SELECT ITEM OR [RETURN] FOR MENU

Esto indica que debe seleccionarse una de las letras que se anteponen a las opciones, para ejecutarla, o bien presionar [RETURN] para volver al MENU.

Cualquiera opción puede ser interrumpida en cualquier momento presionando la tecla [BREAK].

En tal caso el computador volverá a mostrar el mensaje:

SELECT ITEM OR [RETURN] FOR MENU

Si en cualquier momento se presiona la tecla [RESET], el DOS pierde el control del computador.

Si el cartridge de lenguaje BASIC está instalado (ATARI 400 o 800) o está incorporado en el computador (línea ATARI XL), entonces el control pasa al BASIC y aparecerá en pantalla la palabra READY.

Si el cartridge no está instalado o si se presiona [RESET] teniendo la tecla [OPTION] presionada, volverá a cargarse el DOS y el MENU del DOS aparecerá en pantalla.

Las opciones del MENU son:

A. DISK DIRECTORY

Esta opción permite averiguar qué hay en el diskette. Al presionar A aparecerá en pantalla el mensaje:

DIRECTORY -- SEARCH SPEC, LIST FILE?

Si se presiona [RETURN] en respuesta a este mensaje, el computador buscará todos los nombres de archivos en el diskette colocado en la U. de Diskette #1 y los mostrará en pantalla. La lista de todos los nombres de archivos constituye el DIRECTORIO del diskette.

Si el diskette que se está usando es el que contiene el DOS 2.0S, el DIRECTORIO contendrá por lo menos los archivos DOS.SYS y DUP.SYS.

Cada archivo se lista en una línea separada. Hay 4 partes que considerar en cada línea:

a) La marca de protección. Es un asterisco (*) que aparece delante del nombre y que indica que el archivo está protegido. Un archivo protegido así no puede ser alterado ni borrado. Para hacerlo hay que desprotegerlo. Otras opciones del DOS permiten proteger y desproteger un archivo. Si el asterisco no aparece delante del nombre el archivo, no está protegido.

b) El nombre del archivo aparece a continuación, ocupando un máximo de 8 caracteres.

c) Enseguida viene la extensión del nombre con un máximo de 3 caracteres. Nótese que en pantalla no aparece el punto que separa el nombre de la extensión. Tenga el nombre del archivo 8 caracteres o no, la extensión siempre empezará en la novena posición.

d) Por último, a continuación de la extensión se encuentra un número que indica el tamaño del archivo en sectores, es decir, en bloques o grupos de 128 bytes.

La última línea del DIRECTORIO señala el número de sectores libres que quedan disponibles en el diskette.

Cuando el diskette está lleno, es decir, cuando no queda en él mas espacio para guardar información, el computador dá un mensaje de ERROR de DISCO LLENO (ERROR-162).

- LISTAR DIRECTORIO DE CUALQUIER UNIDAD DE DISKETTE.

Si se tiene más de una Unidad de Diskette y se quiere obtener el directorio del diskette colocado, por ejemplo, en la U. de Diskette #2, entonces, como respuesta al mensaje:

DIRECTORY -- SEARCH SPEC, LIST FILE?

responda con:

D2: [RETURN]

La D de D2 puede ser omitida: (2: [RETURN])

- LISTADO RESTRINGIDO DEL DIRECTORIO

Si se desea que aparezcan en pantalla sólo una parte de los archivos contenidos en el diskette, por ejemplo:

i) los que empiezan con la letra R, como respuesta al manejo, debe escribirse:

R*.*

ii) los archivos del diskette de la U. de Diskette #2 que tienen la extensión BAS:

D2:*.BAS

- LISTADO DEL DIRECTORIO EN CUALQUIER PERIFERICO

Como se habrá observado, el mensaje:

DIRECTORY -- SEARCH SPEC, LIST FILE?

consta de 2 partes: la primera es SEARCH SPEC que pide la especificación del archivo para poder buscarlo. Esta especificación queda completa cuando se señala la U. de Diskette (D2:, D3:, D4:) y el nombre del archivo con su extensión, si la tiene. Ej.:

D2:PROGRAM1.BAS

La segunda parte del mensaje (LIST FILE) pide el código del dispositivo donde se quiere listar el DIRECTORIO.

Así por ejemplo, si se quiere listar en la impresora:

i) el DIRECTORIO del diskette colocado en la U. de Diskette #1, bastará con responder:

D1:* *, P:

o bien omitir completamente la especificación del archivo y escribir:

P:

ii) el DIRECTORIO del diskette ubicado en la U. de Diskette #2:

D2:*.*, P:
2:*.*, P:
2:, P:

Cualquiera de estas 3 respuestas cumplirá con el objetivo propuesto.

En el DOS 3.0 la opción leer el directorio se obtiene presionando la tecla [F], una vez que se tiene el MENU del DOS en pantalla, respondiendo a los mensajes que muestra el computador. Para mayores detalles vea el Manual de Referencia para el sistema Operativo de Diskette. (ATARI Disk Operating System Reference Manual).

B. RUN CARTRIDGE.

Esta opción pasa el control del computador al cartridge que está instalado en la ranura izquierda del ATARI 800 o en la ranura de cartridge del ATARI 400. Estos cartridge pueden ser de lenguaje (BASIC por ejemplo) o de juegos u otros programas.

En el caso de la serie XL, el lenguaje está incorporado en ROM en el computador mismo. En el caso de no tener ningún tipo de cartridge colocado en la ranura, el control pasará al lenguaje BASIC.

En el DOS 3.0 esta opción se llama TO CARTRIDGE.

C. COPY FILE.

Con esta opción es posible copiar el contenido de un archivo en otro archivo. Estos archivos pueden estar en el mismo diskette, siempre que tengan nombres distintos. Si se desea que los dos archivos tengan el mismo nombre (al fin y al cabo el contenido de ambos es el mismo), debe efectuar la operación COPY desde un diskette a otro diskette distinto, y siempre que amboos diskettes sean accesibles simultáneamente. Para esto son necesarias dos Unidades de Diskettes. Si se cuenta con una sola, y se quiere copiar un archivo de un diskette a otro, debe usarse la opción O, DUPLICATE FILE.

La opción COPY acepta* caracteres "comodines" (wildcards). Debe tenerse en cuenta, sin embargo, que los archivos que tienen extensión SYS no serán copiados si se usan "comodines". Estos archivos deben ser copiados escribiendo el nombre completo cuando el computador pida la especificación correspondiente. Ejemplos:

a) COPY FILE -- FROM, TO
   CIVIL.BAS, CIVIL.BAK

En este caso no hay especificación de Unidad de Disco. El DOS usará la D1: para ambos. Los nombres de los archivos están separados por coma. La copia se hará en el mismo diskette. El archivo CIVIL.BAS será copiado en otro lugar del diskette con el nombre CIVIL.BAK.

b) COPY FILE -- FROM, TO
   *.BAK, D2:

Esto copiará todos los archivos con extensión .BAK desde la diskettera 1 a la 2.

La opción COPY tiene una característica adicional. Se puede copiar un programa o archivo a continuación de otro. Para ello es necesario agregar los caracteres /A a continuación de la especificación del archivo de destino. Esta sub opción se llama APPEND:
Ejemplo:

COPY FILE -- FROM, TO
D2: CARTA1.TXT, D3: CARTA2.TXT /A

Esto copiará CARTA 1 a continuación del archivo CARTA 2 en la Unidad de Disco #3. El archivo CARTA 2 debe existir previamente.

Esta operación será posible siempre que los dos archivos hayan sido grabados en los respectivos diskettes usando el comando LIST. El proceso no se efectuará si se han grabado con SAVE.

En DOS 3.0 la opción aparece como COPY/APPEND. Igual que en el caso de COPY, permite copiar de un disco a otro o agregar un archivo a continuación del otro, copiar en el mismo diskette, etc. Para los detalles de operación refiérase al Manual de Referencia del DOS 3.0.

D. DELETE FILE (S).

Después de algún tiempo es posible que Ud. ya no necesite de uno o más archivos grabados en un diskette y quiera borrarlos. Puede hacerlo. Con la opción D, DELETE FILE (S). Al ingresar la opción D, aparecerá en pantalla el mensaje:

DELETE FILE SPEC

Entonces se debe especificar el archivo que se desea borrar (número de U. de Disco y nombre del archivo con su extensión, si la tiene). Para especificar varios archivos con algo en común se pueden usar los caracteres "comodines".

Enseguida el computador presentará TYPE "Y" TO DELETE

Después de presionar Y y [RETURN], el DOS buscará todos los archivos que cumplan con la especificación dada y traerá el nombre a pantalla, seguido de un signo de interrogación. Si desea borrar el archivo deberá escribir Y y presionar [RETURN].

Si hay algún archivo protegido,se interrumpirá el proceso y aparecerá ERROR-167. No se puede borrar archivos protegidos. Es necesario desprotegerlos previamente.

Para borrar todos los archivos de un diskette especifique:

.

Cuando se usan "comodines" para especificar los archivos, deben agregarse los caracteres /N si no se desea confirmar la decisión de borrarlos. Ejemplo:

DELETE FILE SPEC
./N

Sin embargo hay que tener cuidado con esta opción, porque los archivos se borrarán irrevocablemente.

En el DOS 3.0 esta opción se llama ERASE.

E. RENAME

En la eventualidad que desee cambiar el nombre de un archivo, es decir, ponerle otro nombre, se debe usar la opción E, RENAME.

Al ingresar E, aparece el mensaje:

RENAME - GIVE OLD NAME, NEW

Ingrese entonces el nombre antiguo del archivo, una coma y el nuevo nombre.

Se puede especificar el número de la U. de Diskette para el nombre antiguo, pero no para el nuevo.

Si el nombre antiguo no existe se producirá ERROR-170 y si el archivo está protegido, ERROR-167. Ejemplo:

```
RENAME - GIVE OLD NAME, NEW
PROGRAMA.BAS, PROGRA1.BAS
```

Esto cambiará el nombre del archivo PROGRAMA.BAS ubicado en la U. de Diskette 1, por el nombre PROGRA1.BAS.

Esta opción también permite el uso de "comodines". Ejemplo:

```
RENAME - GIVE OLD NAME, NEW
*.LET, *.TXT
```

Todos los archivos cuya extensión es LET, cambiarán a archivos cuya extensión será TXT.

Hay que tener mucho cuidado al renombrar archivos. Se debe evitar que el nuevo nombre ya exista en otro archivo del diskette. Si esto ocurre tendrá 2 archivos con el mismo nombre y así cualquier referencia que haga a uno de ellos afectará al otro. Si Ud. vuelve a renombrar uno también renombrará al otro. La única manera de salir del problema es borrar los dos y comenzar de nuevo.

No se deben cambiar los nombres de los archivos que tienen la extensión SYS. (Por ejemplo: DOS.SYS o DUP.SYS. De otro modo el diskette no cargará o no se podrá acceder al MENU del DOS).

En DOS 3.0 la opción RENAME tiene el mismo nombre.

F. LOCK FILE

Esta opción permite proteger archivos contra cualquier acción que signifique alterar la información almacenada en el archivo.

El mensaje que aparece al ingresar F, es:

WHAT FILE TO LOCK?
■

Como respuesta debe especificarse el número de la Unidad de Diskette, si es diferente de 1, y el nombre del archivo que se quiere proteger.

Ejemplo:

D3:CARTA 5.LET

protegerá el archivo CARTA 5.LET ubicada en la Unidad de Diskette #3.

Esta opción también admite el empleo de los caracteres "comodines".

La especificación *.BAS dejará protegidos todos los archivos de la U. de Diskette #1 que tengan la extensión BAS.

Es buena idea proteger los archivos del DOS: DOS.SYS y DOS.DUP.

En DOS 3.0 la opción LOCK FILE aparece con el nombre de PROTECT.

G. UNLOCK FILE

La opción G hace lo contrario de la F. Si un archivo está protegido lo desprotege.

El mensaje que aparece al seleccionar esta opción es:

WHAT FILE TO UNLOCK?
■

Igual que en el caso anterior debe especificarse la U. de Diskette, si es necesario, y el nombre del archivo que se quiere desproteger.

Esta opción acepta el uso de caracteres "comodines".

En DOS 3.0 la opción UNLOCK FILE toma el nombre de UNPROTECT.

H. WRITE DOS FILE

El sistema operativo de discos (DOS) está escrito en 2 archivos: DOS.SYS y DUP.SYS. La opción H permite copiar estos archivos en otro diskette. Para que un diskette cargue automáticamente debe tener por lo menos el archivo DOS.SYS. El DUP.SYS no es necesario si nunca se va a hacer uso de las opciones del MENU cuando se utilice este nuevo diskette. Para eliminar el DUP.SYS, use la opción H y enseguida la opción D para borrar el archivo DUP.SYS. De esta manera los 42 sectores (5376 bytes) que ocupa el DUP.SYS quedarán libres y a disposición del usuario.

El mensaje en pantalla, después de ingresar la opción H, es:

DRIVE TO WRITE FILES TO?
■

Aquí se debe responder con el número de la U. de Diskette: 1, 2, 3, o 4. Supongamos 1.

Enseguida aparecerá un segundo mensaje:

TYPE "Y" TO WRITE DOS TO DRIVE 1

Si se ingresa cualquiera otra letra distinta de Y, la operación quedará anulada.

El DOS. 3.0 no contempla esta opción de manera específica. Durante la operación de la opción INIT DISK, aparecerá un mensaje preguntando: WRITE FMS.SYS FILES?

Si se responde Y, el archivo FMS.SYS se escribirá en el diskette. Con este programa el diskette cargará el DOS 3.0. Sin embargo para poder usar las opciones del MENU, deben incluirse en el diskette los otros dos archivos del sistema (KCP.SYS y KCPOVER.SYS). Estos archivos deben copiarse del diskette del DOS 3.0 al otro diskette usando la opción COPY/APPEND.

I.- FORMAT DISK

Antes de usar un diskette es preciso "formatearlo" (Prepararlo para recibir información). El proceso de "formatear" consiste en que el DOS coloca en el diskette una serie de marcas e información que posteriormente permitira ubicarse dentro de él. Es como si el DOS impusiera en el diskette una plantilla de información básica que posteriormente le permitirá reconocerlo.

La operación de "formateo" es peligrosa. Al "formatear", el DOS borra cualquiera información que contenga el diskette. De este modo se debe estar muy seguro de que el diskette que se quiere "formatear" esté vacío o no contenga información valiosa.

El término "formatear" ha sido incorporado por el uso al lenguaje de los usuarios de computadores.

Para "formatear" un diskette, ingrese la opción I del MENU del DOS 2.0S. En pantalla aparecerá el mensaje:

WHICH DRIVE TO. FORMAT
■

Debe ingresar el número de la Unidad de Diskette en que desea "formatear". En seguida el computador le pedirá que confirme su elección, antes de proceder. Esta confirmación la solicita el computador como una última oportunidad para arrepentirse por si fuera el diskette equivocado el que se va a formatear.

Ejemplo:

WHICH DRIVE TO FORMAT
1
TYPE "Y" TO FORMAT DRIVE 1
■

Esta es la última oportunidad para desistir de formatear. Si ingresa cualquier letra distinta de Y, la operación FORMATEAR se cancelará y el diskette no se borrará. Si ingresa Y el diskette se borrará. A medida de que se ejecuta la operación se escucharán algunos sonidos por el parlante del TV.

En DOS 3.0 la operación FORMAT DISK corresponde a INIT DISK (Inicialización del diskette). Para los detalles de operación, refiérase al Manual de Referencia del DOS 3.0.

J. DUPLICATE DISK

Esta opción permite copiar todo el contenido de un diskette en otro. Puede copiar de una Unidad de Diskette a otra o bien, puede copiar usando una sola Unidad de diskette. Esta opción copia todos los archivos, incluso aquellos que tienen la extensión SYS.

Para duplicar ingrese la opción J. En pantalla aparecerá el siguiente mensaje:

DUP DISK - SOURCE, DEST DRIVES?
■

Si se responde 1,1 la duplicación se hará en una sola U. de Diskette.

Aparecerá entonces un 2º mensaje:

INSERT SOURCE DISK, TYPE RETURN
■

Es el momento de colocar en la U. de Disco el diskette que se quiere duplicar (source disk = diskette fuente) y presionar [RETURN].

El computador leerá parte del contenido del diskette y lo guardará en memoria. En la pantalla aparecerá el mensaje:

INSERT DESTINATION DISK, TYPE RETURN
■

Ahora debe colocarse en la U. de Disco el diskette formateado en el que quiere duplicar la información. Presione [RETURN].

El computador escribirá en el diskette la información que guardó en memoria. En seguida pedirá que coloque nuevamente el diskette fuente, y así sucesivamente. Deberá cambiar de diskette varias veces hasta que el proceso quede terminado, lo que toma normalmente algunos minutos, dependiendo de la cantidad de información almacenada en el diskette fuente. Un valor de referencia es alrededor de 4 minutos.

Si se dispone de 2 disketteras la duplicación será mucho más rápida, sin necesidad de tener que alternar los diskettes fuente y de destino.

En este caso después de tener en pantalla el mensaje:

DUP DISK - SOURCE, DEST DRIVES?
■
la respuesta debe ser:

1, 2

Aparecerá en seguida:

INSERT BOTH DISKETTES, TYPE RETURN

Deberá colocar el diskette fuente en la diskettera 1, el diskette de destino en la diskettera 2 y luego presibnar [RETURN].

En DOS 3.0 la opción DUPLICATE DISK toma el nombre DUPLICATE.

Las opciones K.BINARY SAVE, L.BINARY LOAD y M. RUN AT ADDRESS permiten leer, escribir y ejecutar programas en lenguaje de máquina. Manejan "archivos objeto", que son archivos en números binarios que usualmente son creados usando el ASSEMBLER/EDITOR de ATARI. Para lograr una cabal comprensión de estas funciones es preciso estar familiarizado con el lenguaje assembler o lenguaje de máquina.

N. CREATE MEM.SAV

Para crear el archivo especial MEM.SAV de que hemos hablado anteriormente (ver párrafo 6.6), hay que seleccionar la opción N.

Aparecerá el mensaje:

TYPE "Y" TO CREATE MEM.SAV

Ingrese Y para crear el archivo. Cualquier otra letra o caracter cancelará la operación. Si el archivo ya ha sido creado aparecerá en pantalla el mensaje::

MEM.SAV FILE ALREADY EXISTS

y se cancelará la operación.

O. DUPLICATE FILE

Como se mencionó al tratar la opción C. COPY FILE, esta opción permite copiar un archivo de un diskette a otro, cuando se dispone de una sola diskettera.

6.8 PROGRAMAS EN LENGUAJE BASIC EN DISKETTE

Hay cinco comandos en el lenguaje BASIC que permiten una conexión útil entre el BASIC y el DOS. Los comandos SAVE y LIST permiten almacenar información (programas) en diskette. Los comandos LOAD, ENTER y RUN hacen posible recuperar programas grabados en un diskette.

- GRABACION DE UN PROGRAMA

Al escribir un programa en el computador, éste permanecerá en memoria hasta que sea borrado por un comando tal como NEW o bien hasta que se apague el computador. Los comandos SAVE y LIST permiten guardar el programa en un diskette bajo la forma de un archivo.

Para conseguirlo hay que especificar el número de la U. de Disco en operación, el nombre del archivo y de la extensión, si la hay.

Cualquiera de estos dos comandos grabará un programa en un diskette colocado en la U. de Diskette 1:

```
LIST "D1: CUENTA1.LST"
SAVE "D1: ARCHIVOA.BAS"
```

Si se trata de la U. de Disco 1 se puede omitir el número de la U. de Disco, pero no el nombre de la U. de Diskette (D: es el nombre)

```
SAVE "D: PUZZLE.BAS"
```

permitirá grabar el programa PUZZLE en el diskette que está en la U. de Diskette.

El comando SAVE graba el programa usando abreviaturas (tokens) para las palabras, instrucciones o comandos claves, tales como INPUT, GOTO, PRINT, etc. Estos "tokens" están representados por un solo caracter y representan a los comandos mencionados. Así, por ejemplo, al grabar un programa usando el comando SAVE, la instrucción PRINT que tiene 5 caracteres será grabada representándola con 1 solo caracter (su token).

Por otra parte el comando LIST graba el programa usando el código ATASCII para cada uno de los caracteres que constituyen el programa.

El comando SAVE, siempre graba el programa completo desde la memoria del computador al diskette. El comando LIST puede grabar el programa entero o parte de él. Es necesario especificar líneas entre las cuales se desea grabar el programa.

Por ejemplo, el comando siguiente permite grabar desde la línea 25 a la 150 (ambas inclusive):

```
LIST "D:CUENTA.LST", 25, 150
```

- CARGA DE UN PROGRAMA

LOAD permite cargar desde un diskette al computador aquellos programas que hayan sido grabados usando SAVE. ENTER carga programas que hayan sido grabados usando LIST. Dado que los formatos en que han sido grabados son distintos (SAVE graba usando "TOKENS", LIST graba usando código ATASCII), los comandos LOAD y ENTER no son intercambiables.

Para cargar un programa en memoria debe especificarse el número de la Unidad de Disco, el nombre del archivo y la extensión. Si se omite el número de la Unidad de Disco, se supone que se usa la 1.

LOAD "D:ARTICULO.BAS" carga en memoria el programa ARTICULO.BAS, desde la Unidad de Disco 1. El programa ha sido grabado usando el comando SAVE.

ENTER "D: COMBATE.LST" carga en memoria el programa COMBATE.LST, grabado previamente usando el comando LIST.

El nombre y extensión del archivo usado para cargar el programa debe ser EXACTAMENTE igual al usado para grabarlo. De otro modo el computador no lo encontrará y dará ERROR-170. Cuando el computador encuentra el archivo, lee el programa desde el diskette y al finalizar la lectura aparece en pantalla la palabra READY.

LOAD borra cualquier programa que hubiera en la memoria del computador al cargar un nuevo programa.

El comando ENTER, mezcla el programa existente en memoria con el que se especifica. Si hay números de líneas que son comunes a los 2 programas las nuevas líneas reemplazan a las antiguas. Para eliminar la mezcla de los programas, use el comando NEW antes de emplear el comando ENTER.

Hemos dicho que después de cargar el programa en memoria (con LOAD o ENTER), en pantalla aparece la palabra READY. Sólo falta ejecutar el programa. Para ello hay que escribir RUN y presionar [RETURN].

Sin embargo es posible cargar y correr el programa empleando un solo comando. Si se especifica:

RUN "D:ARTICULO.BAS"

el resultado será que el archivo ARTICULO.BAS se cargará en memoria y a continuación se ejecutará.

El comando RUN "D:ARTICULO.BAS" es una combinación de los comandos LOAD y RUN. Este comando no funciona para los programas grabados con LIST.

6.9 ARCHIVOS DE DATOS EN DISKETTE

La unidad de diskette y el diskette mismo son un dispositivo y un medio excelente para almacenar grandes cantidades de datos. Las instrucciones PRINT # y PUT permiten guardar, grabar o escribir información en el diskette. INPUT # y GET permiten leer o recuperar dicha información.

Desde el punto de vista del computador no hay diferencia entre un programa (archivo de programa) o un archivo que contiene sólo datos (archivo de datos). Para el computador, en último termino ambos archivos son una colección de números. La diferencia está en cómo son interpretados ambos conjuntos de números.

Desde el punto de vista del usuario, un archivo de programa contiene las líneas de una secuencia lógica de instrucciones (programa) y un archivo de datos contiene información, compuesta por valores numéricos y/o alfanuméricos.

En general, un archivo está organizado de una manera lógica. Pensemos, por ejemplo, en un programa tipo base de datos, para tener el historial de los pacientes de un consultorio médico. Cada ficha del paciente con el nombre, domicilio, edad, sexo, síntomas, diagnósticos, etc., constituye un REGISTRO. Cada uno de los items que forman el registro (nombre, domicilio, etc.) es un CAMPO. Estos items o CAMPOS son, normalmente, los mismos para todos los REGISTROS. Lo que varía son los DATOS que llenan los CAMPOS.

Es habitual, entonces, tener un diskette que contiéne- el programa y uno o más diskettes que contienen los archivos de datos, es decir, toda la información relativa a los pacientes. Sin embargo toda esta información es inútil si no se cuenta con el programa, que lleva en sí las instrucciones para manejar y procesar dicha información.

Hay 2 métodos o maneras para trabajar con archivos.

Uno es el método de acceso secuencial o archivo secuencial. Este archivo es similar en su operación a un archivo en cassette. Para leer o escribir el último item del archivo debe leer o escribir todos los items previos.

Los archivos indexados o de acceso de igual probabilidad (random files) son mucho más flexibles. Es posible leer o escribir cualquier registro del archivo con igual facilidad, independientemente de su ubicación.

El DOS 2.0S admite la utilización de ambos tipos de archivos.

En un diskette la información está almacenada en SECTORES. En un diskette formateado con DOS 2.0S caben 720 sectores. Cada SECTOR contiene 128 bytes. El DOS ocupa 13

sectores de cada diskette para almacenar el programa que hace que el DOS se cargue a sí mismo (sectores 1, 2 y 3), para guardar el directorio del diskette (sectores 361 a 368), para tener la información de cuales sectores están en uso y cuales desocupados en el diskette (sector 360). El sector 720 está también ocupado. Por consiguiente quedan libres 707 sectores.

Para facilitar la ubicación de un sector en particular, los 720 sectores van dispuestos en 40 círculos concéntricos con 18 sectores cada uno. Estos círculos se denominan "tracks".

Así la cabeza para leer/escribir de la Unidad de Disco se desplaza hasta el track en que se encuentra la información deseada. De este modo leerá un máximo de 18 sectores para encontrarla.

6.10 ARCHIVOS SECUENCIALES

- 6.10.1 APERTURA DE ARCHIVOS

Para poder usar los archivos, éstos deben haber sido creados previamente. Para crear un archivo o, en general, para operar con archivos, ya sea de programa o de datos, es necesario abrir uno o más canales de información. También se usa la expresión abrir un archivo, que realmente significa abrir un canal para crear un archivo, escribir en él o leer la información contenida en él.

Al abrir un archivo se reserva un espacio de memoria para ser usada como "buffer" de archivo (file buffer). Buffer es un espacio de la memoria dedicado al almacenamiento temporal de datos.

Este "buffer" de archivo es similar al buffer de disco (disk buffer), pero está dedicado a un archivo específico.

El comando OPEN abre un canal de entrada/salida dirigido hacia un archivo.

Por ejemplo:

```
OPEN #2, 8, 0, "D1:ARCHIV01.BAS"
```

Este comando abre el canal 2 para salida hacia el archivo ARCHIVO1.BAS, en la diskettera 1.

Recordemos la estructura del comando OPEN:

a) El primer parámetro indica el número del canal.

Los canales 1 a 5 están siempre disponibles para usarlos en programas en BASIC. Los canales 6 y 7 pueden ser también usados con ciertas precauciones. Los modos gráficos del BASIC usan el canal 6. Los comandos CLOAD, CSAVE y LPRINT usan el canal 7. Si se emplea cualquiera de estos comandos o los niveles gráficos en cuestión, automáticamente harán uso de los canales 6 o 7.

Después que el programa abre un canal hacia un archivo determinado, las referencias se prosiguen empleando el número del canal y no el nombre del archivo. Por esta razón la instrucción OPEN debe ubicarse en el programa antes de que se haga cualquiera otra referencia al archivo.

b) El segundo parámetro, determina el tipo de modo que se va a usar:

LEER (INPUT)	- 4
ESCRIBIR (OUTPUT)	- 8
ACTUALIZAR (UPDATE)	- 12
AGREGAR (APPEND)	- 9
LEER DIRECTORIO (DIRECTORY INPUT)	- 6
ACTUALIZAR CON AGREGADO	- 13 (Sólo para DOS 3.0. Ver Manual DOS 3.0)

c) En este caso se ignora el tercer parámetro de OPEN. Debe ponerse cero (0).

d) Finalmente, el último parámetro especifica el dispositivo externo o periférico hacia el cual se ha abierto el canal. En el caso que nos ocupa, es la Unidad de Diskette. Debe especificarse el número de la Unidad de diskette y el nombre del archivo. Si no se especifica, el computador usará la número 1. Para designar la Unidad de Diskette se usa D:. Puede especificarse cualquier Unidad de diskette, nombre de archivo o extensión.

Si se abre un archivo para cualquier modo que no sea el 8 (output-escribir), éste debe haber sido creado previamente, es decir, debe existir. De otra manera se producirá ERROR - 170.

El modo 8 es el único que permite crear un archivo. Si el archivo ya existe, y se usa el modo 8, el archivo será creado nuevamente, es decir, primero se borrará el existente y luego se creará un nuevo archivo con el mismo nombre pero sin datos almacenados en él.

En DOS 2.0S se puede abrir un máximo de tres archivos simultáneamente, usando, por supuesto, tres canales distintos.

Puede modificarse con POKE, para abrir hasta 7 archivos.

6.10.2 CIERRE DE ARCHIVOS DE DATOS

La mejor manera de cerrar un canal que ha sido abierto hacia un archivo en diskette es mediante la instrucción CLOSE y el mejor momento para hacerlo es apenas se haya terminado de usarlo.

Esto es especialmente cierto cuando se trata de archivos que han sido abiertos para enviar información (output) al diskette. El no cerrar los canales respectivos puede provocar pérdida de datos. Ejemplo:

```
25 PRINT "ABRIREMOS ARCHIVO"
30 OPEN #1, 8, 0 "D:ARCHIVO1.TXT"

50 PRINT "EL ARCHIVO YA ESTA ABIERTO"
60 CLOSE #1
70 END
```

6.10.3 ESCRIBIR EN UN ARCHIVO

La información es enviada al archivo del diskette mediante la instrucción PRINT #, especificando el número del canal que está abierto hacia el archivo en que se desea escribir.

Así, usando el ejemplo anterior:

```
25 PRINT "ABRIREMOS ARCHIVO"
30 OPEN #1, 8, 0, "D:ARCHIVO1.TXT"
50 PRINT "EL ARCHIVO YA ESTA ABIERTO"
60 PRINT #1; "RECUERDE EL ALMA DORMIDA"
70 PRINT #1; "AVIVE EL SESO Y DESPIERTE"
80 CLOSE #1
90 END
```

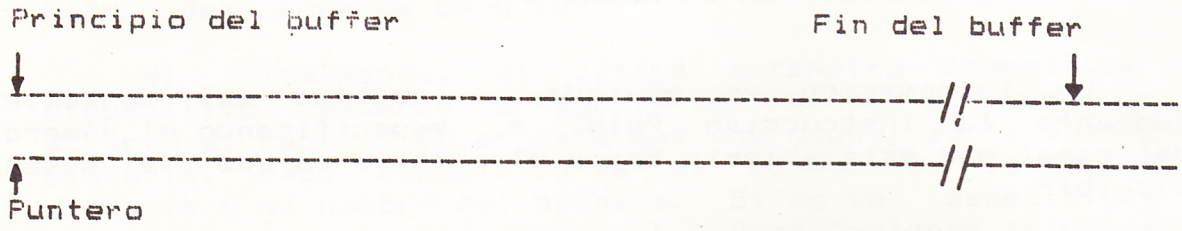
Al correr el programa, el antiguo archivo ARCHIVO1.TXT, es borrado (se está usando modo 8) y vuelto a crear.

La instrucción OPEN coloca un puntero (pointer) al principio del buffer de archivo (buffer file). La primera instrucción PRINT # envía 24 caracteres al archivo (línea 60). Estos caracteres se alinean en el buffer de archivo. El puntero se ubica en la posición siguiente, es decir, en la posición 25. Dado que la línea 60 no termina ni con coma ni con punto y coma, la instrucción PRINT # envía además un caracter EOL (end of line), fin de línea: CHR$(155). El puntero se mueve a la posición 26.

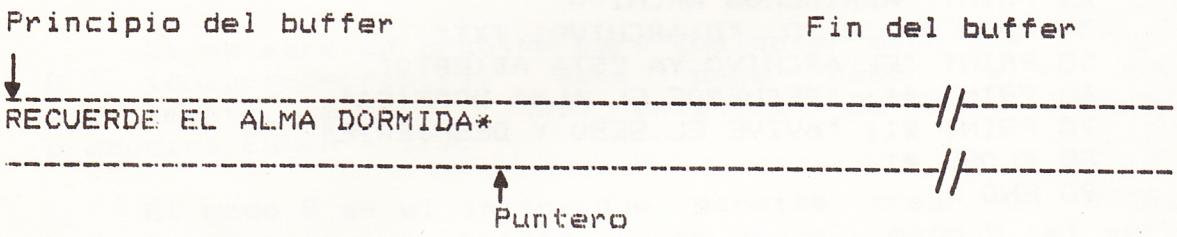
La otra instrucción PRINT # (línea 70) manda al archivo 25 caracteres más un caracter EOL, los que se ubican a continuación del EOL proveniente de la instrucción anterior (línea 60). El puntero queda así en la posición 52.

El programa de ejemplo, ha almacenado 2 campos de información (datos) en el buffer de archivo. Cada campo termina con un caracter EOL.

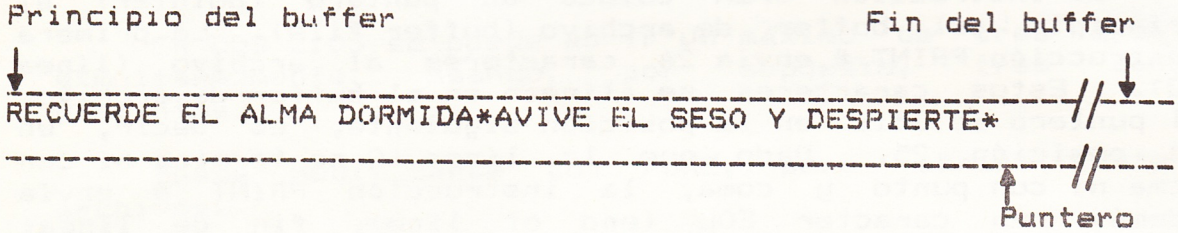
Lo anterior resulta más fácil de entender si se grafica la secuencia explicada (fig. 6-1): EOL=*



a) Buffer de archivo vacío.



b) Buffer de archivo con datos (1 campo).



c) Buffer de archivo con 2 campos.

Fig. 6-1

La instrucción OPEN abre el canal para enviar información y crea un buffer de archivo. Las instrucciones PRINT# envían los datos al archivo, almacenándolo temporalmente en el buffer de archivo.

La instrucción CLOSE hace que el contenido del buffer de archivo sea dirigido hacia el canal 1. Puesto que el canal 1 está a su vez ligado al archivo ARCHIVO1.TXT, el contenido del buffer se escribe en el diskette en el ARCHIVO1.TXT.

El buffer de archivo ocupa 128 bytes de memoria, pero destina 3 para fines de control, quedando 125 bytes disponibles para guardar los caracteres que constituyen la información. Cada vez que el buffer de archivo se llena, su contenido es traspasado al diskette.

En archivos secuenciales, como el presente, el puntero de archivo sólo puede moverse hacia adelante. La única manera de hacer retroceder al puntero es cerrar el archivo (CLOSE) y luego volver a abrirlo. Cada vez que se abre un archivo, el puntero se ubica en el primer caracter del archivo.

Es recomendable experimentar con el programa indicado más arriba, para almacenar otros datos. Cambie las instrucciones PRINT#. Intente usar variables numéricas y alfanuméricas. Use constantes. Agregue más instrucciones PRINT #, siempre que las incorpore entre las instrucciones OPEN y CLOSE.

Un punto importante a recordar es que si cualquier instrucción PRINT # termina con una coma o con un punto y coma, no enviará un caracter EOL al buffer de archivo. Los caracteres correspondientes al siguiente PRINT # formarán parte del mismo campo.

Ejemplo:

```
10 OPEN # 2, 8, 0, "D2:EJEMPLO"
20 PRINT # 2; "ESTE ES EL PRIMERO";
30 PRINT # 2; "Y ESTE EL SEGUNDO CAMPO"
40 CLOSE # 2
50 END
```

En el ejemplo anterior la instrucción PRINT # de la línea 20 es concatenada con el PRINT # de la 30, resultando así los dos textos en un solo campo (fig. 6 - 2):

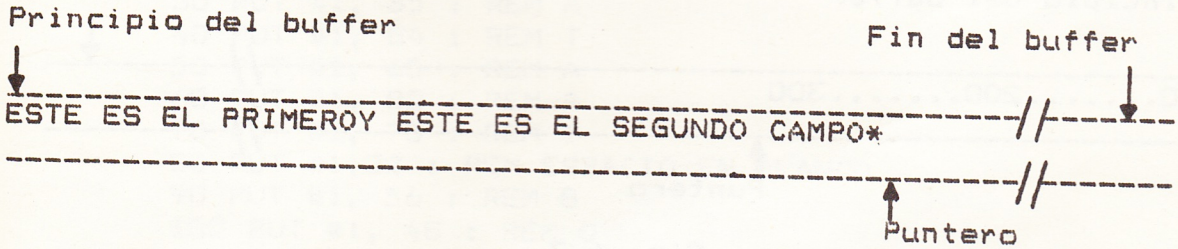


Fig. 6-2

Esto es lo que sucede con el punto y coma. Pero, qué sucede si al final de la instrucción PRINT # se coloca una coma?.

Pués, sucede lo mismo que cuando se escribe información en pantalla: los primeros caracteres quedan separados por nueve espacios. Recuérdese que la coma coloca los caracteres o textos en columnas que tienen un ancho de 10 caracteres.

El computador no se da cuenta de la diferencia entre escribir en pantalla o escribir en un archivo de diskette. Desde su particular punto de vista es lo mismo.

La coma suprime la aparición del caracter EOL.

En el caso de los archivos en diskettes, el puntero del buffer de archivo hace las veces de cursor.

Supongamos que el Canal 3 ha sido abierto para escribir a un archivo de diskette:

```
325 PRINT #3; 100, 200, 300,
```

Esta instrucción enviará un campo de datos al archivo en el diskette. Puesto que los datos o items están separados por coomas, se agregará 7 caracteres en blanco a cada item y, puesto que hay una coma después del último, no se enviará el caracter EOL. (Fig. 6-3).

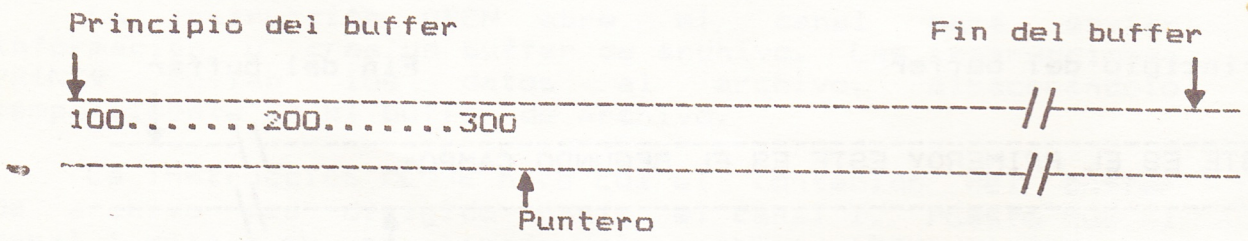


Fig. 6-3

No es aconsejable el uso de comas cuando se opera con PRINT #, a causa de los espacios en blanco que intercala. Si se desea concatenar items es preferible usar punto y coma.

Si no se desea concatenar items, cada campo debe ir en un PRINT # separado. Otra solución es generar explícitamente un caracter EOL, como en el ejemplo siguiente:

```
500 PRINT #5; "DISCULPE"; CHR$ (155); "DIGAME LA HORA,
POR FAVOR"
```

CHR$ (155) genera un caracter EOL.

Si se usa la técnica anterior en forma intensiva, vale la pena asignar el caracter EOL a una variable string.

```
10 DIM E$ (1)
20 E$ = CHR$ (155) : REM CARACTER EOL
,
,
,
150   PRINT#   4;   "NOMBRE";   E$;   "DIRECCION";   E$;
   "TELEFONO"; E$; "RUT #"
```

La instrucción **PUT** también permite escribir datos en un archivo. Sin embargo está limitada a valores numéricos entre 0 y 255. El valor numérico es interpretado como un caracter en código ATASCII. Cada valor ocupa el mismo espacio en el archivo que un caracter salido de un PRINT#. La instrucción PUT no genera por sí misma un caracter EOL.

El programa siguiente ilustra el uso de PUT: guarda en el archivo TITULOS.BAS, el texto "ATARI 800" incluyendo las comillas y terminando con un caracter EOL.

```
10 OPEN #1, 8, 0, "D:TITULOS.BAS"
20 PUT #1, 34 : REM"
30 PUT #1, 65 : REM A
40 PUT #1, 84 : REM T
50 PUT #1, 65 : REM A
60 PUT #1, 82 : REM R
70 PUT #1, 73 : REM I
80 PUT #1,32 : REM ESPACIO EN BLANCO
90 PUT #1, 56 : REM 8
100 PUT #1, 48 : REM 0
110 PUT #1, 48 : REM 0
120 PUT #1, 34 : REM"
130 PUT #1, 155 : REM (EOL)
140 CLOSE #1
150 END
```

6.10.4 LEER UN ARCHIVO

Una vez que la información ha sido grabada en un archivo, es posible recuperarla o leerla desde el archivo. Las instrucciones INPUT # y GET permiten leer los datos del archivo y asignar los valores a las variables.

Veamos como trabajan estas instrucciones. Consideremos el programa siguiente que crea un archivo PRUEBA2.BAS.

Escriba:

```
10 OPEN #1, 8, 0, "D: PRUEBA2.BAS"
20 PRINT #1; "QUEREMOS LEER UN ARCHIVO"
30 PRINT #1; "COMO DEMOSTRACION"
40 CLOSE #1
50 END
```

El archivo PRUEBA2.BAS ya está creado. Hagamos el programa para leerlo:

```
10 DIM A$ (100)
20 OPEN #1, 4, 0, "D:  PRUEBA2.BAS": REM  ABRE  ARCHIVO
PARA LEER
30 INPUT #1; A$
40 PRINT A$
50 GOTO 30
60 CLOSE #1
70 END
```

Al correr el programa anterior, en pantalla aparecerá:

```
QUEREMOS LEER UN ARCHIVO
COMO DEMOSTRACION

ERROR - 136 AT LINE 30
```

El error 136 se produce porque al ejecutar por tercera vez la línea 50 e ir a la línea 30, intenta leer más allá del registro FIN DE ARCHIVO. El error detiene el programa de modo que la instrucción 60 (CLOSE #1) no se ejecuta. En este caso no es importante. El programa no escribió nada en el archivo, (se abrió para leer), de modo que el buffer de archivo no contiene ni nunca contuvo datos que necesitaran escribirse en el archivo.

Sin embargo, no es buena práctica dejar que los programas terminen con ERROR o dejando archivos abiertos, aún cuando no importe.

El programa puede mejorarse agregando una instrucción: (línea 25))

```
10 DIM A$ (100)
20 OPEN #1, 4, 0, "D:PRUEBA2.BAS"
25 TRAP 60
30 INPUT #1; A$
40 PRINT A$
50 GOTO 30
60 CLOSE #1
70 END
```

Ahora, cuando se produzca el ERROR, la ejecución del programa continuará en la línea 25 y de ahí a la 60. El programa terminará bien y no habrá mensaje de ERROR.

El método que dá plena seguridad en la ejecución del programa, consiste en chequear si el error que se produce es realmente el ERROR - 136 (de fin de archivo). En el programa anterior cualquier error provocará el mismo efecto.

A continuación mostramos una versión del programa que verifica el tipo de error.

```
10 DIM A$ (100)
20 OPEN #1, 4, 0, "D:  PRUEBA2.BAS": REM ABRE ARCHIVO
   PARA LEER
25 TRAP 52
30 INPUT #1; A$
40 PRINT A$
50 GOTO 30
52 ERR = PEEK (195): REM OBTIENE NUMERO DEL ERROR
54 IF ERR = 136 THEN 60 : REM FIN DE ARCHIVO?
56 PRINT "HAY ERROR _ "; ERR
60 CLOSE #1
70 END
```

La instrucción **INPUT#** lee un campo de datos cada vez. Realmente lee caracteres, hasta encontrar un caracter EOL. Entonces asigna el valor de lo leído (valor del campo) a la siguiente variable en la lista de variables. Por consiguiente debe existir correspondencia entre el tipo de variable (numérica o alfanumérica) y el tipo de datos (numéricos o alfanuméricos). De otro modo se producirá un ERROR.

La instrucción **GET** lee un archivo caracter por caracter. Cada instrucción GET lee un valor numérico. En el programa debe indicarse cómo interpretar dicho valor numérico. Lo usual es usar la función CHR$ para interpretarlo como código ATASCII.

El programa a continuación utiliza la instrucción GET para leer el archivo PRUEBA2.BAS que hemos usado en el programa anterior.

```
10 DIM A$ (100)
20 OPEN #1, 4, 0, "D:PRUEBA2.BAS"
25 TRAP 52
30 GET #1, A
40 PRINT CHR$ (A);
50 GOTO 30
52 ERR = PEEK (195)
54 IF ERR = 136 THEN 60
56 PRINT "HAY ERROR _"; ERR
60 CLOSE #1
70 END
```

6.10.5 AGREGAR INFORMACION A ARCHIVOS

Cuando se abre un archivo para escribir información (modo 8), el archivo existente se borra y el puntero se ubica al comienzo del nuevo archivo.

Existe la posibilidad de agregar información a la que ya se encuentra almacenada en el archivo. Para ello debe especificarse el modo 9. Trabajando en este modo el puntero se ubica al final del archivo. Es necesario que el archivo exista o se producirá ERROR 170.

Veamos un programa que muestre como trabaja el modo 9. Primero creemos un archivo EJEMODO9.BAS:

```
10 OPEN #2,8,0,"D: EJEMODO9.BAS"
20 PRINT #2; "AGREGAREMOS A ESTE _";
30 PRINT #2; "ARCHIVO LO SIGUIENTE:"
40 CLOSE #2
100 END
```

Una vez creado el archivo, debemos agregarle información empleando el modo 9:

Escribamos a continuación del programa anterior:

```
50 OPEN #3, 9, 0, "D:EJEMODO9.BAS"
60 PRINT #3; "MODO 9 EN DEMO"
70 CLOSE #3
```

Al usar el modo 9 (línea 50) para abrir el archivo, el puntero del buffer de archivo queda colocado para seguir escribiendo, al final del archivo existente (fig. 6-4)

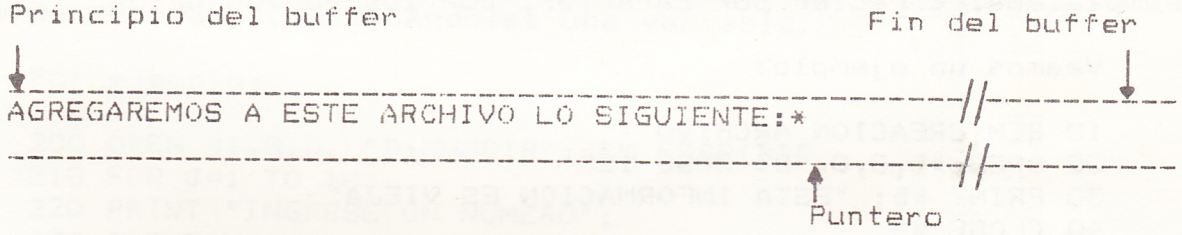


Fig. 6-4

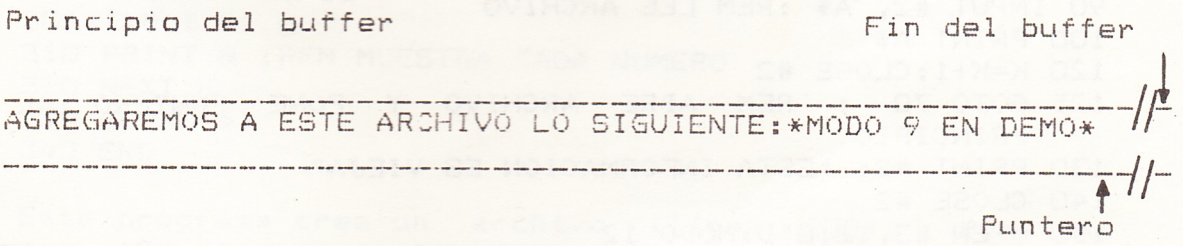


Fig. 6-5

Cada vez que se usa el modo 9 (APPEND.AGREGAR), se asigna un nuevo sector (128 caracteres) al archivo.

Hay otra manera de agregar información a un archivo.

Los modos 8 y 9 sólo permiten escribir: el modo 4 sólo permite leer. Existe un modo, el 12, que permite hacer ambas cosas.

Este modo 12 del que hablamos es llamado ACTUALIZAR o PONER AL DIA (UPDATE).

Cuando se abre un archivo para actualizar información, el puntero del buffer de archivo se ubica al comienzo del archivo y el contenido del archivo permanece inalterado. Entonces uno puede leer o escribir en el archivo (usando la instrucción apropiada). Por cada caracter que se lea o se escriba el puntero del buffer de archivo se moverá un lugar hacia adelante.

Si se elige escribir, la información existente será reemplazada, caracter por caracter, por los nuevos datos.

Veamos un ejemplo:

```
10 REM CREACION ARCHIVO
20 OPEN #5,8,0,"D: MODO 12"
30 PRINT #5; "ÈSTA INFORMACION ES VIEJA"
40 CLOSE #5
50 DIM A$ (30)
60 K=0
65  REM  ABRE  ARCHIVO  PARA  ACTUALIZAR.  PUNTERO  AL
   PRINCIPIO
70 OPEN #2,12,0,"D:MODO 12"
80 IF K=1 THEN 130
90 INPUT #2, A$ :REM LEE ARCHIVO
100 PRINT A$
120 K=K+1:CLOSE #2
125 GOTO 70  :  REM  ABRE  ARCHIVO  Y  PONE  PUNTERO  AL
    PRINCIPIO
130 PRINT #2; "ESTA INFORMACION ES VIEJA"
140 CLOSE #2
150 OPEN #3,12,0"D:MODO 12"
160 INPUT #3, A$
170 PRINT A$ :REM ESCRIBE  EN  PANTALLA  LA  INFORMACION
    ACTUALIZADA
180 CLOSE # 3
190 END
```

6.10.6 LOS NUMEROS EN LOS ARCHIVOS

Hay varias maneras de guardar números en un archivo. El siguiente programa muestra algunos:

```
10 PRINT "ABRIR ARCHIVO"
20 OPEN #2,8,0,"D:NUMEROS"
30 PRINT "ARCHIVO ABIERTO"
40 PRINT #2; "VIVO EN PROVIDENCIA 1270"
50 PRINT #2;15,25,35,45
60 A=10 : B=20 : C=30 : D=40 : E=50
70 PRINT #2;C,D,E,A + B,C * D
80 CLOSE #2
90 END
```

En la línea 40, los números se almacenan como parte de un valor alfanumérico. También pueden guardarse directamente, ya sea como constantes numéricas (línea 50) o bien usando variables previamente asignadas (línea 70).

La instrucción INPUT # permite leer directamente los números guardados, asignándoles una variable.

Por ejemplo:

```
200 OPEN #1,8,0, "D:NUMDIR":REM ESCRIBIR
210 FOR J=1 TO 10
220 PRINT "INGRESE UN NUMERO";
230 INPUT N
240 PRINT #1; N :REM GUARDA CADA N EN NUMDIR
250 NEXT J
260 CLOSE #2
270 OPEN #1,4,0,"D:NUMDIR":REM LEER
280 PRINT "UD. INGRESO:"
290 FOR K=1 TO 10
300 INPUT #1,N
310 PRINT N :REM MUESTRA CADA NUMERO
320 NEXT K
330 CLOSE #1
340 END
```

Este programa crea un archivo (línea 200). Pide se ingresen 10 números cualesquiera y los guarda en el archivo NUMDIR (líneas 210 a 260). La línea 260 cierra el archivo

y la 270 lo reabre, con lo cual el puntero del buffer de archivo se ubica al comienzo del archivo, listo para leer. El programa lee los números y los muestra en pantalla (líneas 290 a 320).

Es importante saber como se leen los números desde el archivo en diskette.

Desde el punto de vista del computador, un número es todo lo que hay dentro de un campo. Es decir lo que hay hasta que encuentra un caracter de fin de línea (EOL). Si encuentra una coma, leerá el número hasta ese caracter, pero seguirá leyendo información adicional y descartándola hasta que encuentre un caracter EOF. Ninguno de los caracteres que encuentre entre la coma y el EOF tiene algún efecto sobre el valor numérico que se ha leído. (EOF = END OF FILE = FIN DE ARCHIVO).

Asi por ejemplo, la instrucción PRINT# puede almacenar más de un número, cuando los números están separados por coma.

```
715 PRINT# 5; 2, 4, 6,
```

En este caso no hay caracter EOL y el efecto de la coma es insertar espacios en blanco, como se mencionó anteriormente (ver Fig. 6-3).

Se puede intentar leer tales valores con una variable numérica, de a uno por vez con:

```
800 INPUT # 5; A
810 INPUT # 5; B
820 INPUT # 5; C
```

Pero no resultará.

Tampoco resultará con una instrucción como la siguiente:

```
850 INPUT # 5; A, B, C
```

En cualquiera de estos casos se producirá un error, por cuanto los espacios en blanco producidos al grabar (línea 715) no están permitidos en la definición de números.

Si en la instrucción PRINT# (línea 715) se usa punto y coma en vez de coma para separar los números, entonces éstos quedarán concatenados y al intentar leerlos serán interpretados como un solo número.

La manera de evitar estos problemas es asegurarse que cada valor quede separado por un caracter EOL.

Consideremos el ejemplo siguiente:

```
10 OPEN #2,8,0,"D: DATOSNUM"
20 FOR K=1 TO 10
30 PRINT #2;K;
40 NEXT K
50 CLOSE #2
```

Puesto que la instrucción PRINT # termina con un punto y coma, no hay EOL y los 10 valores se almacenarán como un solo gran número. En este caso el problema se soluciona eliminando el punto y coma al final de la línea 30.

Se puede almacenar números separados por punto y coma, pero debe incluirse una coma entre cada valor. La coma debe ser colocada entre comillas, de la manera siguiente:

```
315 PRINT # 4; 125; ","; 300; ","; 12
```

Hay que tener cuidado de leer los datos en la misma forma en que se guardaron; en este caso como un conjunto de 3 números.

```
400 INPUT # 4; P, Q, R
```

Las siguientes instrucciones: no son válidas:

```
410 INPUT # 4; P
420 INPUT # 4; Q
430 INPUT # 4; R
```

En este caso el segundo y tercer número serán descartados, mientras el BASIC busca un caracter EOL.

6.11 ARCHIVOS DE ACCESO DIRECTO (RANDOM ACCESS FILES)

En el archivo secuencial es necesario leer primero todos los registros ubicados antes del registro deseado. Una vez encontrado el registro, por este método, es posible acceder a la información.

El acceso directo permite hacer referencias o dirigirse a cualquier parte del archivo independientemente del resto del archivo. Las instrucciones del BASIC que permiten el acceso directo son NOTE y POINT. Estas instrucciones no funcionan con el sistema operativo 1.0 pero sí con el 2.0S y 2.5, de la manera descrita a continuación.

Con el DOS 3.0 las instrucciones operan de modo distinto. Consulte su manual de DOS 3.0.

La instrucción NOTE se usa para determinar la ubicación del puntero de archivo. El computador presenta esta ubicación como 2 parámetros: el número del último sector a que se ha hecho referencia y el número de ubicación del último caracter en referencia, dentro de aquel sector. Por ejemplo:

```
25 NOTE $ 2, A, B
```

Esta instrucción se refiere al archivo hacia el cual se ha abierto el canal 2. Asigna a la variable numérica A el número del último sector al que se ha accedido. Asigna, además, a la variable B el número del último caracter a que se ha accedido en ese sector.

El número del sector que entrega el computador no es un número relativo al número de sectores en el archivo de que se trata, sino que es el número absoluto del sector en el diskette. Puede ser cualquier número de 1 a 719 (DOS 2. OS). El primer sector del archivo no tiene que ser el número 1, ni el segundo número el 2. De hecho, en general no lo son. Así el primer sector podría ser el 548 y el segundo el 325 o 147.

```
150 A = 35 : B = 77 : POINT #1,A,B
```

La instrucción **POINT** es la opuesta de NOTE.POINT, permite ubicar el puntero de archivo en el sector y caracter que el programa especifica. Una instrucción posterior tal como PRINT #, PUT, INPUT # o GET permitirá escribir o leer en ese archivo a partir de la ubicación (sector y byte) especificada. Por ejemplo:

Con esta instrucción el puntero del archivo abierto para el canal 1, es llevado al sector 35 del diskette y dentro de ese sector al bite número 77.

Los números de sector y byte deben ser especificados por variables numéricas. Deben ser variables que hayan sido definidas anteriormente. El canal al que se refiere el símbolo #, debe haber sido previamente abierto para el archivo en cuestión. El número del sector (A), debe estar comprendido entre 1 y 719 y el número del caracter (B), entre 0 y 125 (que es el número de bytes disponibles dentro de un sector).

Hasta el momento en que deba efectuarse la lectura o escritura no se comprueba si el sector especificado está o no dentro del archivo para el que se ha abierto el canal.
Cuando esto ocurra el DOS verificará que el sector a que se accede sea parte del archivo especificado por el número de canal. Si el resultado de la verificación es negativo, se producirá uno de estos errores:

- ERROR 170 (ARCHIVO INEXISTENTE). Error de lectura.
- ERROR 171 (INSTR. POINT NO VALIDA). Error de escritura.

6.11.1 ARCHIVO INDEXADO

Un método muy usado para trabajar con archivo directo es el de archivo indexado. En este método se usan dos archivos en lugar de uno; uno mantiene la información correspondiente a cada registro que necesita utilizar el programa, el otro es un índice del primer archivo. El segundo archivo contiene una clave o código para cada uno de los registros del primer archivo. Junto con cada código el archivo índice guarda la ubicación del comienzo de cada registro de datos del primer archivo. Cada una de estas ubicaciones está especificada por el número del sector y el número del caracter.

Al comienzo del programa, los códigos y sus números de sector y caracter asociados son leídos y guardados en arreglos (arrays).

Para encontrar un registro, el programa busca en el archivo índice hasta que encuentra el código del registro requerido. Luego, utiliza los números de sector y caracter asociados para ubicarse en esas coordenadas (con la instrucción POINT), que son las del comienzo del registro deseado. Enseguida puede leer el registro de datos con INPUT # o GET o escribir en él, con PRINT # o PUT.

En los archivos indexados se puede agregar un registro entre los ya existentes; es un proceso demoroso, aunque no muy complicado. Hay que mover los registros para hacerle espacio al nuevo. En el archivo índice hay que agregar el nuevo código y modificarlo de tal manera que todos los códigos que queden después del nuevo, apunten hacia las ubicaciones correctas.

Para borrar un registro se requiere un trabajo similar, pero en sentido contrario. Hay que mover todos los registros que siguen al que se quiere borrar, eliminar el código del registro borrado del archivo índice y cambiar los punteros para todos los códigos que están a continuación de la clave eliminada.

Para ahorrar memoria, el programa puede guardar el número del sector y del caracter en una variable string. Las funciones CHR$ y ASC permiten hacer la conversión entre string y valores numéricos. Se necesitan sólo tres caracteres para definir el código de cada registro: 2 para el número de sector (1 a 719) y 1 para el número del caracter (0 a 125).

La mayor utilidad y conveniencia de los archivos indexados reside en la rapidez con que un registro puede ser ubicado. Buscar en un arreglo de códigos en memoria (en el ATARI BASIC el arreglo queda simulado por una gran string dividida en substring), es una operación que se realiza con mucha mayor rapidez que la búsqueda a través de la lectura registro por registro de un archivo.

6.11.2 EJEMPLO DE ARCHIVO INDEXADO

El programa a continuación presenta un ejemplo de manejo de archivo indexado: La línea 10 dimensiona las variables alfanuméricas que se usarán en el programa. A$ es la string que guardará los datos del registro en forma de sub-strings. TC$ es la string de códigos que guardará en forma de sub-strings los códigos y los números de sector y caracter asociados. Está dimensionada para aceptar hasta 100 códigos de hasta 4 caracteres. C$ es la string que permite almacenar los 7 caracteres correspondientes al código (4) y a los números de sector y caracter (3) que permiten ir a buscar directamente el registro deseado en el archivo de registros.

Los archivos de registro y código se inicializan en la línea 100 y en la línea 130 se vuelven a abrir para ingresar los datos (modo 9).

En la línea 320 se realiza la grabación de los registros (canal 2) y de los códigos respectivos (canal 1).

La línea 310 contiene la instrucción NOTE que toma nota de la ubicación del puntero de archivo, antes de proceder a la grabación.

Como los archivos están trabajando en el modo 9 (agregar información), el puntero se coloca al final de cada archivo antes de grabar.

La fórmula que aparece a continuación de la instrucción NOTE # 2 A, B (línea 310), permite modificar el número de sector y caracter leído por NOTE para grabarlo en la línea siguiente (320). El número del sector leído es expresado en módulo 155, es decir, el número del sector, de 1 a 719 en DOS 2.0S o de 1 a 1010 en DOS 2.5, será expresado comos dos números, ambos menores que 155, en código ATASCII.

Por ejemplo, si el número del sector leído es el 513 y el número del caracter es el 55:

K= INT (513/155) = INT (3.30967741) = 3

P= 513 - 3 * 155 =48

C$ (5,5) = CHR$ (3) = = CTRL - C

C$ (6,6) = CHR$ (48) = 0 = CERO

C$ (7,7) = CHR$ (55) = 7

Así entonces, los 3 caracteres que definen la ubicación del registro en el archivo de datos serán 07.

En la línea 390 y siguientes se leen los archivos; la línea 450 indica la manera cómo el computador busca la dirección del registro, mediante la decodificación de la ubicacion dada por los punteros de sector y caracter. Es el proceso inverso que el anteriormente descrito.

ARCHIVOS INDEXADOS

```
10 DIM A$(51),T$(34),OP$(1),TC$(700),CO$(6),C$(7)
20 GRAPHICS 0:POKE 710,80:? :? "                MENU ARCHIVO DIRECTO"
25 ? "          ----------------------":? :?
30 ? :? :? "          1 INICIALIZA ARCHIVOS"
40 ? :? :? "          2 INGRESO DE DATOS":? :? :? "     3 CONSULTA DE DATOS":?
50 ? :? :? "                DIGITE OPCION: ";:INPUT O
60 POSITION 24,17:? "";:INPUT O
70 ON O GOTO 95,110,360
90 REM INICIALIZA ARCHIVO ABRIENDO CON "OPEN" OPCION 8
95 POKE 710,0:? CHR$(125):POSITION 8,12:? "INICIALIZANDO ARCHIVOS"
100 OPEN #1,8,0,"D:CODIGO.DAT":OPEN #2,8,0,"D:REGIS.DAT":CLOSE #1:CLOSE #2:GOTO 20
110 REM INGRESO DE NUEVOS REGISTROS
120 REM APERTURA DE LOS ARCHIVOS
130 TRAP 11000:OPEN #1,9,0,"D:CODIGO.DAT":OPEN #2,9,0,"D:REGIS.DAT"
140 GOSUB 640:POKE 710,194:? CHR$(125):? "          CREACION    "
145 FOR C=0 TO 37:? "_";:NEXT C
150 GOSUB 160:GOTO 210
160 ? " 1.CODIGO      = "
170 ? " 2.NOMBRE      = "
180 ? " 3.DIRECCION   = "
190 ? " 4.FONO        = "
200 RETURN
210 POSITION 18,3:INPUT T$:C$(1,4)=T$
220 POSITION 18,4:INPUT T$:A$(1,20)=T$
230 POSITION 18,5:INPUT T$:A$(21,40)=T$
240 POSITION 18,6:INPUT T$:A$(41,51)=T$
250 GOSUB 290:POSITION 4,18:? "          DIGITE OPCION :";:INPUT OP$
260 IF OP$="M" THEN 210
270 IF OP$<>"C" THEN 250
280 GOTO 310
290 POSITION 2,20:? "-----------------------------------------"
295 ? "    M Modificar  C Continuar":RETURN
300 REM ANOTACION DE DIRECCIONES Y GRABACION
310 NOTE #2,A,B:K=INT(A/155):P=A-K*155:C$(5,5)=CHR$(K):C$(6,6)=CHR$(P)
320 $(7,7)=CHR$(B)
330 PRINT #1;C$:PRINT #2;A$
340 CHR$(125):POSITION 4,10:? "DESEA CONTINUAR S/N:";:INPUT OP$
350 OP$="S" THEN 140
360 OP$<>"N" THEN 330
370 OSE #1:CLOSE #2:CLOSE #3:GOTO 20
```

```
360 REM MANTENCION (CONSULTA Y MODIFICACION)
370 REM ABRIR ARCHIVOS PARA MODIFICAR
380 OPEN #1,12,0,"D:CODIGO.DAT":OPEN #2,12,0,"D:REGIS.DAT"
390 REM LECTURA REGISTRO DE CLAVES Y DIRECCIONES
400 TRAP 410:FOR C=1 TO 100:NOTE #1,A,B:INPUT #1;C$:TC$(C*7-6,C*7)=C$(1,7)
405 CON=CON+1:NEXT C
410 POKE 710,228:? CHR$(125):? "DIGITE CODIGO ARTICULO(F=FIN):";:INPUT CO$
415 IF CO$="F" THEN CLOSE #1:CLOSE #2:GOTO 20
420 IF CO$="" THEN CO$="       "
430 FOR C=1 TO CON:C$="     ":C$(1,LEN(CO$))=CO$:TRAP 460
440 REM BUSCANDO LA DIRECCION DEL REGISTRO
450 IF C$=TC$(C*7-6,C*7-3) THEN 465
460 NEXT C:GOSUB 650:GOTO 410
465 S=ASC(TC$(C*7-2))*155+ASC(TC$(C*7-1)):B=ASC(TC$(C*7,C*7))
470 SW=0:POINT #2,S,B:INPUT #2;A$
480 ? CHR$(125):? "       MANTENCION  ":? :GOSUB 160
490 POSITION 19,3:? C$:POSITION 19,4:? A$(1,20)
500 POSITION 19,5:? A$(21,40)
510 POSITION 19,6:? A$(41,51)
520 OP$="1":POSITION 2,20:? "----------------------------------------"
525 ? " RETURN PARA SEGUIR (M=MODIFICA):";:INPUT OP$
530 IF OP$<>"M" THEN 620
540 TRAP 540:SW=1:POSITION 2,20:? "----------------------------------------"
545 ? CHR$(156);CHR$(156);"     Digite Nro Item (2-4):";:INPUT O
550 IF O<2 OR O>4 THEN 540
560 O=O-1:ON O GOSUB 590,600,610
570 GOTO 520
590 POSITION 18,4:INPUT T$:A$(1,20)="                    "
595 A$(1,20)=T$:RETURN
600 POSITION 18,5:INPUT T$:A$(21,40)="                    "
605 A$(21,40)=T$:RETURN
610 POSITION 18,6:INPUT T$:A$(41,51)="           ":A$(41,51)=T$:RETURN
620 IF SW=0 THEN 410
630 POINT #2,S,B:PRINT #2;A$:GOTO 410
640 FOR C=1 TO 51:A$(C,C)=" ":NEXT C:C$="                    ":RETURN
650 POKE 752,1:FOR R=1 TO 5:FOR L=1 TO 20:POSITION 10,10
655 ? "CODIGO NO EXISTE":NEXT L
660 FOR L=1 TO 20:POSITION 10,10:? "CODIGO NO EXISTE"
665 NEXT L:NEXT R:POKE 752,0:RETURN
```

6.11.3 ARCHIVOS INDEXADOS CON DOS 3.0

El sistema operativo de diskette DOS 3.0 tiene en sus primeras versiones dos anomalías que es bueno corregir.

Para determinar si necesita corregir su versión del DOS 3.0, cargue el DOS 3.0 (sin presionar OPTION) y ejecute el siguiente comando:

PRINT PEEK (1816)

Si el valor que entrega el computador es "53" el DOS 3.0 en examen es una versión actualizada y no requiere ser modificada.

Si el valor entregado es "51" o "56", el DOS 3.0 en cuestión debe ser modificado, con el programa FIX.BAS que se entrega más adelante.

Las 2 anomalías que requieren corrección son:

1.- Agregar información a un archivo que tiene 64513 bytes. Si el archivo tiene entre 64513 y 65537 bytes y se desea agregar información, al abrir el archivo en modo 9, se produce un ERROR 177: Estructura del Diskette dañada. Es imposible agregar información al archivo.

2.- Usando el comando POINT con el DOS 3.0, resulta imposible acceder a una ubicación mayor que 32767 bytes desde el comienzo del archivo. Si se hace un intento en este sentido se producirá el ERROR 7: Número mayor que 32767 y el comando POINT no se completará.

Estos problemas fueron corregidos en una versión posterior del DOS 3.0 de ATARI. Sin embargo, subsistió un error de I/O al intentar elegir una opción desde el MENU HELP, o al querer usar la función MEM SAVE.

El programa FIX-BAS corrige todos estos problemas.

USO DEL PROGRAMA DE CORRECCION FIX.BAS.

Para crear una copia corregida del DOS 3.0 de ATARI proceda de la siguiente manera:

1.- Duplique el diskette maestro de DOS 3.0 usando la opción DUPLICATE DISK del MENU.

2.- Cargue el DOS 3.0 duplicado en memoria con el cartridge ATARI BASIC en el computador: 800 antiguo.

3.- Escriba el programa FIX-BAS en pantalla y córralo escribiendo RUN y presionando RETURN. Es aconsejable guardar este programa en diskette antes de correrlo, para el caso en que se produzca algún problema al ejecutarlo, o bien para el caso en que se desee usarlo para corregir otro diskette.

ES MUY IMPORTANTE que FIX-BAS sea copiado correctamente antes de ser ejecutado. Si cualquiera de las instrucciones DATA es mal copiada, PUEDEN CREARSE ERRORES GRAVES EN LA COPIA QUE SE PRETENDE CORREGIR.

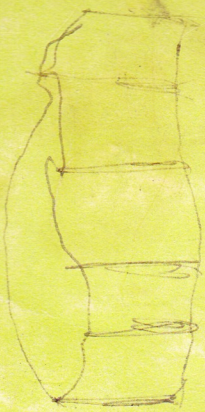
4.- Después de corregir la copia del DOS 3.0, coloque el autoadhesivo de protección contra escritura. Este diskette será ahora su MASTER DISKETTE. Uselo para crear otros diskettes de trabajo.

El programa FIX BAS, es el siguiente:

```
10 REM FIX.BAS
20 REM FIX ATARI DOS 3 ANOMALIES
30 REM
40 DIM FILE$(16)
50 PRINT CHR$(125):REM CLEAR SCREEN
60 IF PEEK(2117)=240 THEN FLAG=1
70 REM FLAG FOR CORRECTED FMS.SYS
80 REM
90 REM
100 READ FILE$
110 IF FILE$="DONE" THEN 280
120 OPEN #1,12,0,FILE$
130 READ CHANGES
140 PRINT "CORRECTING ";CHANGES;" BYTES IN- ";FILE$
150 FOR I=1 TO CHANGES
160 READ OFFSET,BYTE
170 IF BYTE=76 THEN CHECK=77
180 IF BYTE=77 THEN CHECK=78
190 IF BYTE=78 THEN CHECK=76
200 IF BYTE=240 THEN CHECK=208
210 IF FLAG=0 THEN HI=OFFSET:LO=0:GOSUB 330
220 IF FLAG=1 THEN GOSUB 320
230 IF ERROR=1 THEN GOTO 300
240 NEXT I
250 CLOSE #1
260 ? "CORRECTION TO ";FILE$;" COMPLETED":PRINT
270 GOTO 100
280 ? "ALL CORRECTIONS COMPLETED"
290 END
300 ? "ERROR -- THE FILE , ";FILE$;" DOES NOT CONTAIN THE EXPECTED BYTES"
310 ? :? "PLEASE CREATE A NEW WORK DISKETTE AND BEGIN AGAIN":CLOSE #1:STOP
320 HI=INT(OFFSET/256):LO=OFFSET-HI*256
330 POINT #1,HI,LO:GET #1,TEST
340 IF BYTE=53 OR BYTE=52 OR BYTE=49 OR BYTE=48 THEN GOTO 380
350 IF TEST()CHECK THEN ERROR=1
360 IF FILE$="D:FMS.SYS" AND TEST=BYTE THEN ERROR=0:RETURN
370 IF ERROR=1 THEN RETURN
380 POINT #1,HI,LO:PUT #1,BYTE
390 RETURN
400 REM
410 DATA D:KCPOVER.SYS,3,304,78,309,76,312,77
420 DATA D:HELP.UTL,3,1568,76,1572,78,1577,77
430 DATA D:CONVERT.UTL,9,3106,78,3111,76,3146,76,3152,78,3158,78,3164
    DATA 78,3167,76,7173,76,3486,77
440 DATA D:COPY.UTL,5,3656,78,3661,76,3955,77,4065,78,4071,76
450 DATA D:FMS.SYS,14,28,52,30,53,31,48,32,49
    DATA 331,240,2103,78,2109,76,2115,77,2137,78,2143,76,2149
    DATA 77,2157,78,2163,76,2169,77
460 DATA DONE
```

CODIGOS DE ERROR

CODIGO DE ERROR	MENSAJE DE ERROR CORRESPONDIENTE
2	Memoria insuficiente
3	Error de Valor Numérico
4	Demasiadas Variables
5	Largo de Variable alfanumérica Excedido
6	Falta de Datos (DATA)
7	Número mayor que 32767 (Número de Línea)
8	Error en INPUT
9	Error en Dimensión de Variable
1Ø	Expresión Matemática demasiado Compleja
11	Superación de la capacidad de manejo de números muy grandes o muy pequeños.
12	Línea inexistente
13	NEXT sin FOR
14	Línea demasiado Larga
15	Línea GOSUB o FOR suprimida
16	RETURN sin GOSUB
17	Error de Sintaxis
18	Variable Alfanumérica No Válida
19	Programa demasiado Largo para ser Cargado
2Ø	Número de Canal Ø o mayor que 7 (OPEN)
21	Error de Carga de un Programa
128	Operación de ENTRADA o SALIDA interrumpida con BREAK
129	Canal abierto previamente
13Ø	Aparato Inexistente
131	Intento de Entrada por un canal de Salida
132	Error de Sintaxis en un comando X10
133	Canal No Abierto previamente
134	Canal de Salida Desconocido
135	Intento de Salida por un canal de Entrada
136	Fin de Archivo
137	Fichero Truncado
138	Aparato No Responde
139	Aparato No Funciona o No puede ejecutar la instrucción
14Ø	Error en la comunicación entre el aparato y el computador
141	Cursor Fuera de Pantalla
142	Inconsistencia en la comunicación Aparato/Computador
143	Error en la Grabación o Lectura desde Cassette o Diskette.
144	Intento de Escritura en un Diskette Protegido
145	Error de Modo de Gráfico (GRAPHICS) o Verificación.
146	Función No Implementada
147	Memoria Insuficiente para Modo Gráfico
16Ø	Número de Unidad de Disco Desconocido
161	Demasiados Archivos Abiertos



Manual de Basic Avanzado Para Computadores

versión atari basic